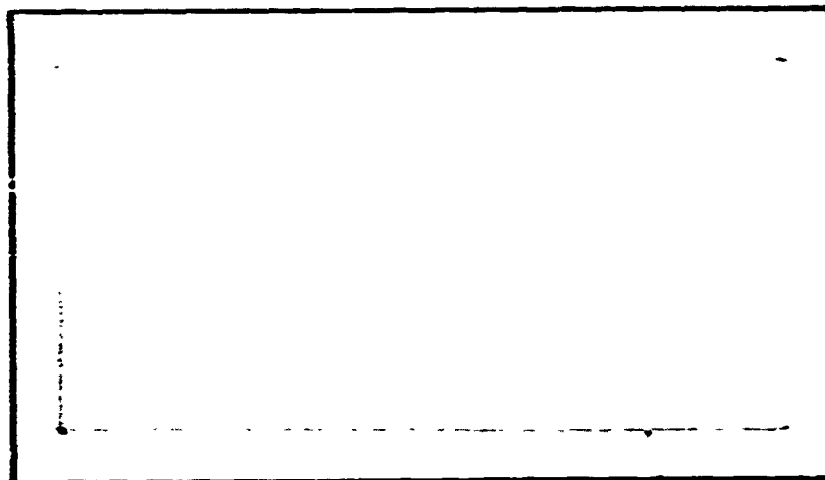


AD742965

AIR FORCE INSTITUTE OF TECHNOLOGY



**AIR UNIVERSITY
UNITED STATES AIR FORCE**



SCHOOL OF ENGINEERING

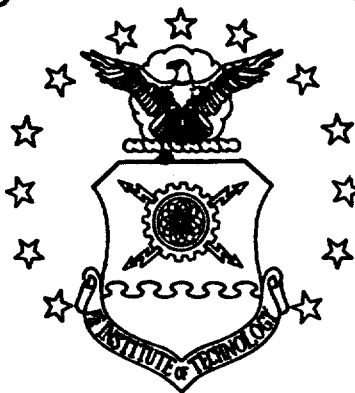
Exponsored by
**NATIONAL TECHNICAL
INFORMATION SERVICE**
Springfield, Va. 22151

WRIGHT-PATTERSON AIR FORCE BASE, OHIO

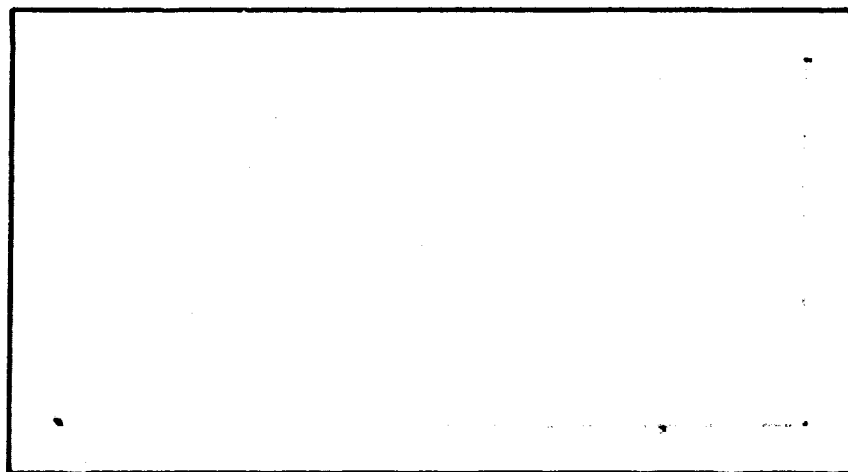
Handwritten: RDC
RECEIVED
31 MAR
1954
Handwritten: R

AD742965

AIR FORCE INSTITUTE OF TECHNOLOGY



AIR UNIVERSITY
UNITED STATES AIR FORCE

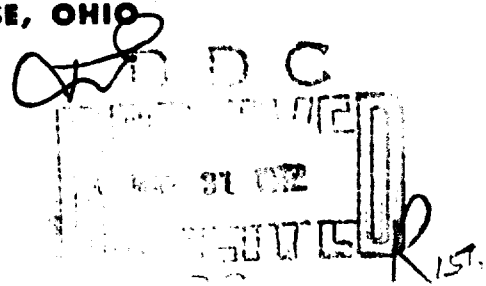


SCHOOL OF ENGINEERING

Reproduced by
NATIONAL TECHNICAL
INFORMATION SERVICE
Springfield, Va. 22151

WRIGHT-PATTERSON AIR FORCE BASE, OHIO

Best Available Copy



Unclassified

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered from the overall report if classified)

1. ORIGINATING ACTIVITY (Corporate author)		2a. REPORT SECURITY CLASSIFICATION	
Air Force Institute of Technology (AFIT-EN) Wright-Patterson AFB, Ohio 45433		Unclassified	
2b. GROUP			
3. REPORT TITLE			
Software Simulation of the Minuteman D17B Computer			
4. DESCRIPTIVE NOTES (Type of report and inclusive dates)			
AFIT Thesis			
5. AUTHOR (Last name, middle initial, first name)			
Bruce Chatterton Captain USAF			
6. REPORT DATE		7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
March 1972		155	8
8a. CONTRACT OR GRANT NO.		9a. ORIGINATOR'S REPORT NUMBER(s)	
A. PROJECT NO. H/A		GE/FE/72-7	
c.		10. OTHER REPORT NUMBERS (How other numbers may be assigned this report)	
d.			
11. DISTRIBUTION STATEMENT			
This document has been approved for public release and sale; its distribution is unlimited.			
12. APPROVAL FOR PUBLIC RELEASE: LAW APR 190417		13. CONTAINS MILITARY ACTIVITY	
Stephen B. Ingram, Captain, USAF Director of Information, AFIT			
14. ABSTRACT			
<p>A software program has been written which simulates the functions of the Minuteman D17B computer at the register transfer level. The simulation program is written in the FORTRAN Extended language to be used on the Intercon System (teletype) of a CDC 6600 computer system. The simulation program consists of a main program and eight subroutines. A programming language for the D17B simulation was forced which contains numbers and load codes, switches, and miscellaneous commands. Example programs run on the simulated computer have been included to show the types of output available.</p>			

DD FORM 1473

Unclassified

Security Classification

Unclassified

Security Classification

14.	KEY WORDS	LINK A		LINK B		LINK C	
		ROLE	WT	ROLE	WT	ROLE	WT
	Computer Simulation Digital Computer Simulation Languages FORTRAN Programming D17B Computer						

Unclassified

Security Classification

SOFTWARE SIMULATION OF THE

KINERMAN D173 COMPUTER

THESE

GE/E2/72-7 Bruce Chatterton
Captain USAF

This document has been approved for public
release and sale: its distribution is unlimited.

D D C
DECEMBER
1972

**SOFTWARE SIMULATION
OF THE
MINUTEMAN D173 COMPUTER**

THESIS

**Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology**

Air University

**in Partial Fulfillment of the
Requirements for the Degree of**

Master of Science

by

**Bruce Chatterton, B.S.E.E.
Captain USAF**

Graduate Electrical Engineering

March 1972

**This document has been approved for public
release and sale: its distribution is unlimited.**

Preface

The contents of this thesis represent the results of doing a software simulation of the Minuteman D17B Computer. The D17B computer is a general-purpose computer which was used in the control of the Minuteman Missile. This computer is being phased out of Air Force inventory, and as a result of being declared excess, it is being made available to government agencies and educational institutions. The Air Force Institute of Technology Electrical Engineering Department has acquired two of these computers.

Research has been started at AFIT to make the D17B computer operational in a laboratory environment and to develop applications. The software simulation is a part of this research effort. The other areas being pursued at the present time are the design and construction of a hardware control console, the design and construction of an I/O interface for controlling a tape reader, tape punch, and teletype, and a description of the D17B computer and the steps to be followed in making it operational.

I want to express my appreciation to Dr. Frank R. Brown and Dr. Gary B. LaMont for proposing the simulation program as an area of research and for their expertise as advisors for this project. Special acknowledgement is due Bob Hitchell, a Systems Engineer from Newark Air Force Station, Ohio, for the knowledge and documentation which he has imparted to this research project. I am also

GE/EE/72-7

grateful to the other four students who were doing research
in this area for their help in understanding the operation
of the D17B computer.

Bruce Chatterton

Contents

	Page
Preface	ii
List of Figures	vi
Abstract	vii
I. Introduction	1
General D17B Description	2
Thesis Outline	6
II. D17B Computer Simulation Program	8
Reading and Translation Section	15
Non-Compute Mode Section	19
Compute Mode Section	24
Subroutines	24
Subroutine STORE	26
Subroutine LOAD	27
Subroutine UNLOAD	27
Subroutine FLAGSTO	28
Subroutine DISPLAY	28
Subroutine MEMORY	29
Subroutine DISCRET	30
Subroutine INCREASE	30
Simulation Program Variables	31
III. D17B Computer Simulation Language	39
Numbers and Load Codes	39
Switches	41
Timing Signal	42
Power On/Off Switch	42
Initiate Loading Switch	42
Cold-Storage Write Switch	42
Master Reset Switch	43
Discrete Switch	43
Mechanical Input Switch	44
Compute Mode Switch	44
Miscellaneous Inputs and Commands	45
Register and Memory Display	45
Incremental Inputs	47
Discrete Inputs	48
Mode Tracing	49
Execution Specification	50
Setting of Flipflops	51
Initialization	51
Programming Methods	52
Shortened Version of Simulation Language	59

Contents

	Page
IV. Error Detection	62
Error Statements/Causes of Errors	62
V. Programming Examples	66
Example Program Number 1	66
Example Program Number 2	71
Example Program Number 3	77
Example Program Number 4	82
Example Program Number 5	84
Example Program Number 6	87
Example Program Number 7	90
VI. Conclusion	93
Recommendations for Future Study	94
Bibliography	96
Appendix A: Printout of Simulation Program	A-1
Appendix B: D173 Instruction Set and D173 Load Codes	B-1
Appendix C: Figures for Interpreting Binary, Discrete, and Voltage Outputs	C-1
Appendix D: Instructions for Using the Simulation Program at AFIT	D-1
Vita	97

List of Figures

<u>Figure</u>		<u>Page</u>
1	Veitch Diagrams of Operation Codes and Special Instructions	10
2	Veitch Diagrams of Channel Addresses and Flag Store and Load Codes	11
3	Correlation between Minuteman and FORTRAN designation for Sector Track and Multi-word Loops . .	13
4	D17B Computer Simulation Program Flow	14
5	Reading and Translation Section Flowchart	16
6	Switch Interpretation Flowchart	17
7	Miscellaneous Input and Command Flowchart	18
8	Non-Compute Mode Section Flowchart	21
9	Manual Halt Mode Flowchart	22
10	Process Code Mode Flowchart	23
11	Compute Mode Section Flowchart	25
12	Discrete Outputs	C-1
13	Binary Outputs	C-2
14	Voltage Output Scheme	C-3

Abstract

A software program has been written which simulates the functions of the Minuteman D173 computer at the register transfer level. The simulation program is written in the FORTRAN Extended language to be used on the Intercom System (teletype) of a CDC 6600 computer system. The simulation program of the D173 computer was developed at the Air Force Institute of Technology as part of a research effort in making a D173 operational in a laboratory environment. The simulation program has proven itself useful as a teaching aid and can be used for error checking program tapes to be run on the D173 computer. It can also be used as a standard for the hardware version of the computer. The simulation program consists of a main program and eight subroutines. The main program consists of a reading and translation section which reads and interprets input data, a noncompute mode section which implements the loading and interaction functions, and a compute mode section which implements the search, read and write memory, and execute functions. A programming language for the D173 simulation program was formed which contains numbers and load codes, switches, and miscellaneous commands. The miscellaneous commands include such functions as register display, memory display, node tracing, and setting of flipflops. Example programs run on the simulated computer have been included to show the types of output available.

SOFTWARE SIMULATION
OF
THE MINUTEMAN D17B COMPUTER

I. Introduction

The purpose of this thesis is to describe the software simulation program of the Minuteman D17B computer that has been developed. A software simulation of the D17B computer was developed as part of a research effort at the Air Force Institute of Technology. This research effort was concerned with finding useful applications for the D17B computer.

There are several reasons why a simulation of the D17B computer was written. This program can be used in teaching the operation of the D17B computer. It can also be used as backup capability for running D17B programs when the actual computer is not available. The most important reason, however, is that the simulation program can provide error checks for the D17B programs which it executes. The hardware version of the D17B computer has no error checking capability.

The simulation program was written to simulate the D17B computer at the register transfer level. A register transfer approach was used because it allowed the D17B computer to be simulated at the information and data transfer level. Thus it was not necessary to simulate the logic equations required to clear and set each flipflop. Using the register transfer approach also allows for the tracing

of the information flow in the simulated computer as data is loaded and programs executed. With this information tracing capability, the simulation program can be used as a teaching aid.

General D17B Description. The D17B computer is a small, synchronous, serial, general-purpose digital computer. It was designed to be used in airborne control applications and was used in controlling the guidance and operation of a Minuteman Missile. This computer has several important characteristics of which the following are important to an understanding of the simulation program. (Ref 6:5-6)

1. When the D17B computer is executing, all computer operations are controlled by an internally stored program. This stored program can be entered by external input devices (tape reader, teletype, control console switches, etc.).

2. The word length for this computer is 27 bits, of which 24 are used in computation. The remaining 3 bits are spare and synchronizing bits and thus were not needed in the simulation program. For this reason the word length is treated as 24 bits throughout the remainder of this thesis.

3. The memory storage capability consists of a 6000 rpm magnetic disk with a storage capacity of 2935 words of which 2728 are addressable. The contents of memory include 20 cold-storage channels of 128 sectors (words) each, a hot-storage channel of 128 sectors, four rapid access loops (U,F, E,H,) of 1, 4, 8, and 16 words respectively, four 1-word arithmetic loops (A,L,H,I), and two 4-word input buffer

loops (V,R). Cold-storage channels are those memory locations which allow data to be stored only when they are enabled by an external switch. However, data can be read from them at all times. Hot-storage channels can be used for storing and reading of data without an enable switch. A loop consists of a word or group of words which are continually read and stored on the disk as it turns. A 1-word loop would be read and stored each wordtime. For a 4-word loop, each word is read and stored in four wordtimes, an 8-word loop is read and stored in eight wordtimes. A wordtime is the amount of time required to serially read and store the 24 bits of a word. All portions of memory described here have been included in the simulation program.

4. The D17B computer performs computations using the binary number system with negative numbers being represented in two's complement form (sign plus two's complement).

5. The instruction set for this computer consists of 39 instructions. The mnemonic and octal coding for each instruction is given in Appendix B. Also included with the instruction set is the number of wordtimes required for the execution phase of each instruction.

6. The input capability of the D17B computer includes acceptance of detector, discrete, incremental, and character inputs. The detector input sets the DR (detector reset) flipflop to "1" when a true level is put on the detector input line. Discrete inputs are true or false levels on the discrete input lines. Incremental inputs are sampled

inputs that are incrementally added to the input buffer loops (V,R). Character inputs are five bit codes generated by a teletype or tape reader and transferred to the D17B on the character input lines.

7. The outputs that can be realized from the D17B computer are binary, discrete, single character, phase register status, telemetry, and voltage outputs. Binary outputs are computer generated levels of +1 or -1 available on the binary output lines. A discrete output is a true level which is put on one of 28 discrete-output lines. Only one discrete output line can be at the true level at a time. Single character is a computer generated five bit code of the 4 most significant bits of the accumulator plus a parity bit. The character output is made available on output lines for driving a teletype, a tape punch, or some other character coded output device. Phase register status is the condition of the phase register flipflops which is available for monitoring on output lines. Telemetry output is the bit configuration of registers or voltage signals available on output lines for transmission to telemetry equipment. Voltage output is a computer generated analog voltage corresponding to portions of the accumulator contents which is made available on output lines.

8. Special features of the D17B computer include flag store, split-word arithmetic, and minimized access timing. Flag store provides the capability of storing the present contents of the accumulator while executing the

next instruction. Split-word arithmetic is used in performing arithmetic operations on both halves of a split word at the same time. A split word on the D17B consists of 11 bits. Minimized access timing is the placing of instructions and data in memory so that they are available with minimum delay from the disk memory.

In order to have the D17B computer simulation program simulate the actual computer as closely as possible, all of the foregoing characteristics have been included. As a result of this similarity, the simulation program shows promise for usefulness as a standard for an operational D17B computer. By comparing the results of a test program provided as input to both the hardware and software versions, register and instruction execution malfunctions in the hardware version can be detected.

The D17B computer can be loaded with programs and data from punched tapes. The program and data are punched onto the tape by a tape punch and a tape reader is used to enter this information into the D17B computer. The simulation program is extremely helpful in the preparation of these program tapes which are to be read into the D17B computer. The simulation program has the capability of reading the same punched tapes for input data as are used in loading the D17B computer. The simulation program helps in the preparation of program tapes by detecting and locating invalid symbols punched on the tape and by decoding the program instructions. The simulation program also has

the capability to detect addresses (locations in memory) that are out of range of the present program being run. These capabilities have shown the D17B computer simulation program to be very useful.

Thesis Outline. Chapter II of this thesis contains a description of the structure and organization of the simulation program. The functions performed by the main program and subroutines are discussed and a description of the variables used in writing the simulation program is given. Chapter III contains a description of the simulation language which is used as input data for the simulation program. Methods for creating programs to be run on the simulated computer are given and a method for creating a shortened version of the simulation language is presented. Chapter IV contains a listing of the error statements provided by the simulation program. Chapter V contains example programs which have been run on the simulated computer. Several programs are listed which show the types of output that are available from the simulation program. Chapter VI is the concluding chapter and contains recommendations for additions to the simulation program to enlarge its capabilities.

Four appendixes are included with this thesis to provide additional information and clarification to the D17B computer simulation description. Appendix A contains a listing of the simulation program. Appendix B is a compilation of the D17B instruction set and a listing and description of the D17B load codes. Appendix C contains

figures for interpreting the simulation program output results for binary, discrete, and voltage outputs. Appendix D supplies information for using the D17B simulation program at AFIT. Also included in Appendix D is a condensed listing of the simulation language.

The description of the D17B computer simulation program presented in this thesis assumes the reader has a basic knowledge of the D17B computer and the procedures for programming it. No attempt is made to describe the D17B computer or to describe D17B programming methods. For information concerning these areas, the reader should refer to references 1 and 4.

References 4, 5, 7, and 8 are the main sources of information used in writing the simulation program. Reference 4 is a training manual for the D17 computer which describes the functions and operations of the computer. Reference 5 is a collection of figures which show pictorially the D17B functions and operations. Reference 7 is an engineering manual with a function breakdown of the logic equations and timing diagrams of the computer operations. Reference 8 is an Air Force Technical Manual containing all the logic equations implemented on the D17B computer.

II. D17B Computer Simulation Program

This chapter describes the organization and structure of the D17B computer simulation program. In writing the simulation program, the plan was to simulate the actual computer as closely as possible. This close correlation between the actual computer and the simulation program makes it possible for a user to use both the computer and simulation program using only one set of programming techniques. However, there are several areas in the simulation program where a quasi-simulation approach was used. The quasi-simulation approach uses the same register inputs and generates the same results, but the methods of obtaining the results differ.

In preparing to write the simulation program, several computer simulation languages were studied, the predominant one being the Computer Design Language (CDL) developed at the University of Maryland. This language consists of computer elements (register, memory, counters, etc.) and is described in the first five chapters of reference 2. Portions of the D17B computer simulation program were written in CDL, but because of the nonavailability of a CDL compiler, a transformation to the FORTRAN language was made.

The simulation program is written in the FORTRAN Extended Language to be run on the Intercom System (teletype) of a CDC 6600 Computer system. Instructions for using the simulation program at AFIT are contained in Appendix D. Appendix A is a listing of the simulation program.

The D17B computer has several codes and addresses which it decodes and uses in loading and executing a program. Karnaugh Maps (Venn diagrams) of the operation codes, flag store codes, load codes, and channel addresses are shown in Figs. 1 and 2. These codes and addresses appear in the computer in binary form. The operation code is a four bit code used to determine the instruction to be executed. The flag store code is a three bit code which determines where flag store will take place. The load codes are five bit codes used in loading data into the computer. An instruction address is a seven bit code which determines the sector location of the next instruction. The instruction channel address can only be changed by using a transfer (TRA) instruction. A number address is a twelve bit code which consists of a five bit channel designation and a seven bit sector location. Because FORTRAN instructions do not operate on binary data, a correlation between the operation code, flag store code, load code, instruction address, and number address of the D17B computer and a number in the FORTRAN program had to be made. This relationship was made by taking each code or address and changing the binary representation to its equivalent decimal representation. The decimal representation was then used as the designation for the code or address in the FORTRAN program. Included on the diagrams in Figs. 1 and 2 are the binary representation, the quasi-octal representation, and the FORTRAN designation.

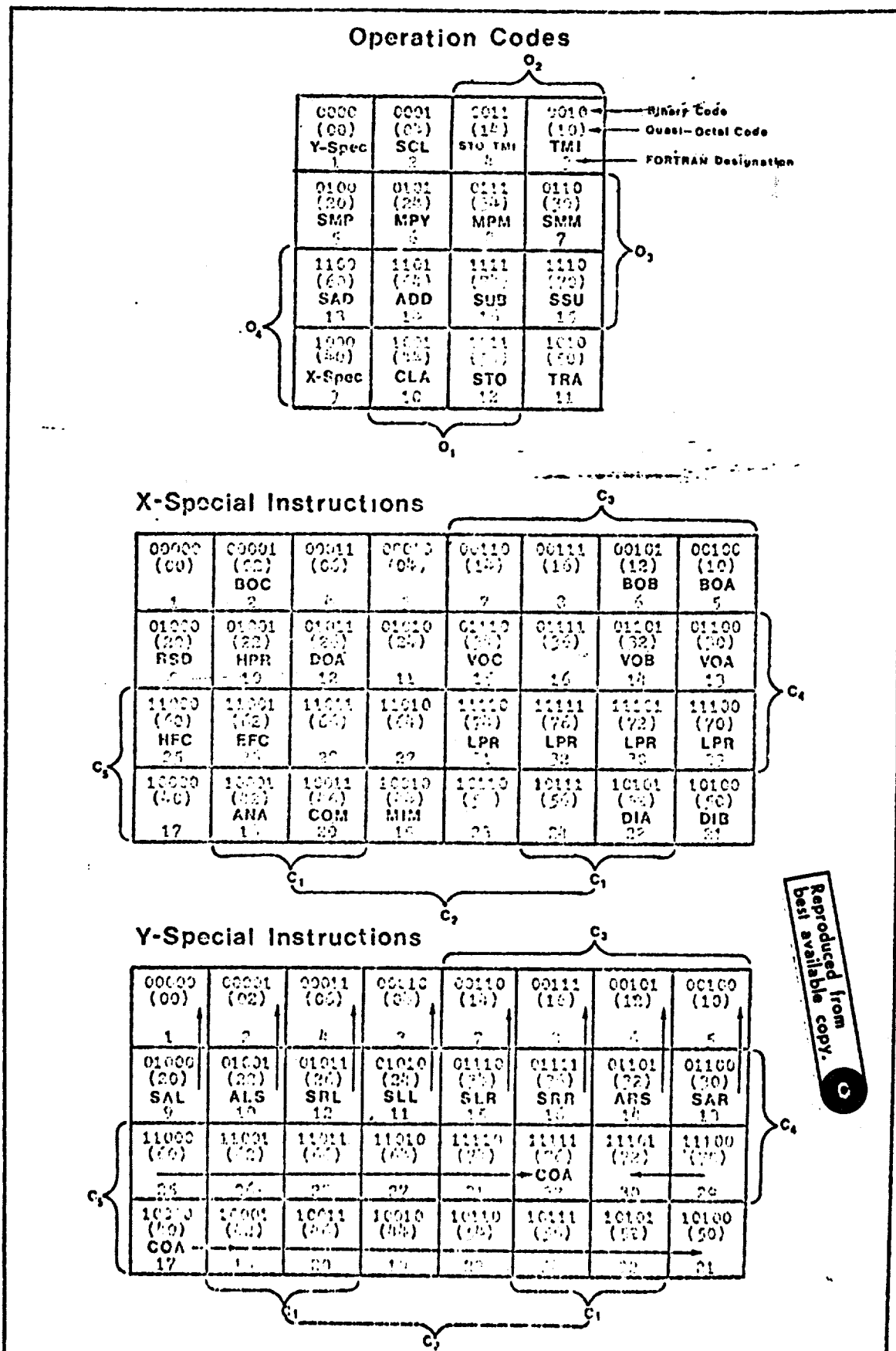


Fig. 1. Veitch Diagrams of Operation Codes and Special Instructions

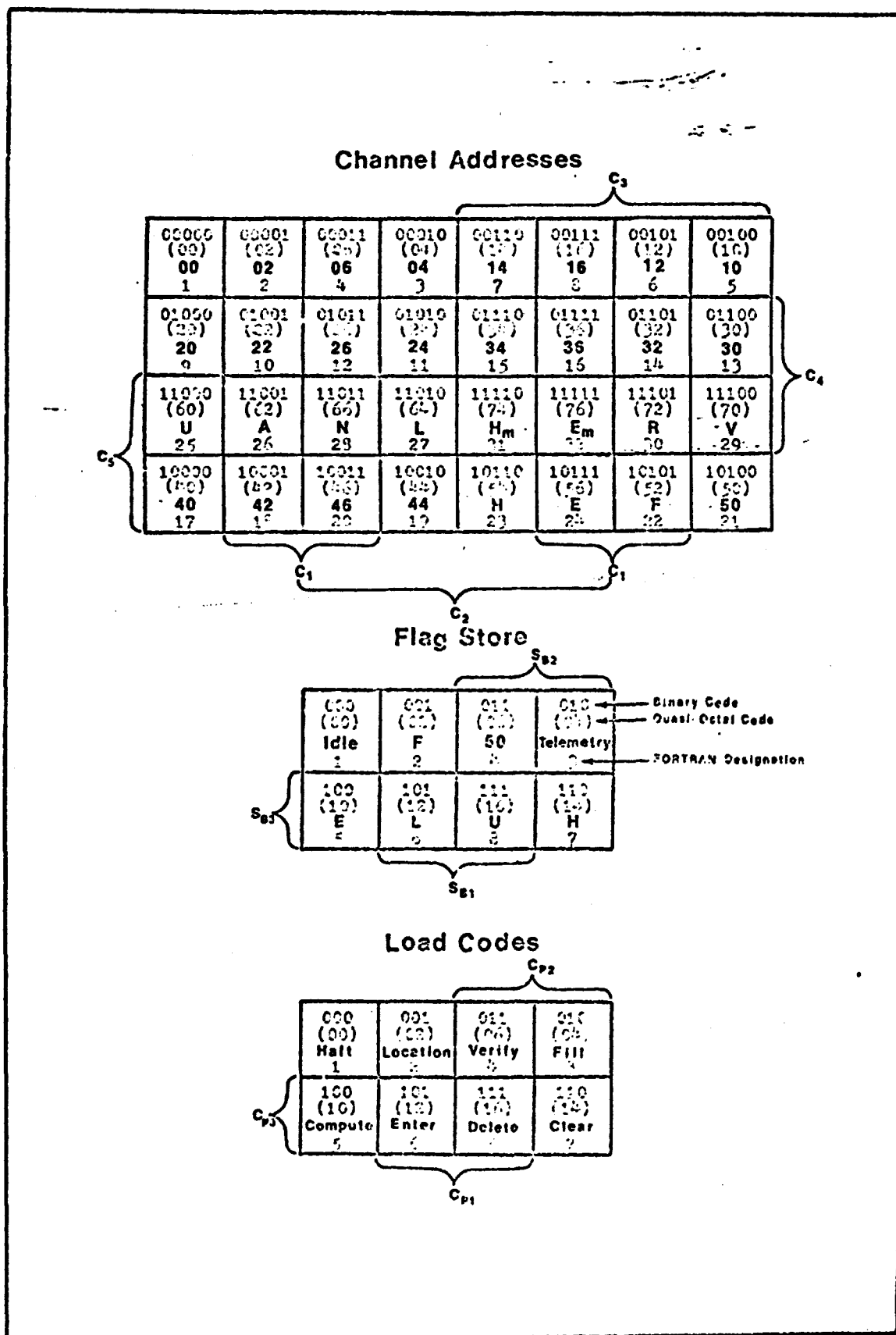


Fig. 2. Veitch Diagrams of Channel Addresses and Flag Store and Load Codes

A quasi-octal representation of the codes and addresses can be made by taking the binary representation and converting to octal. It is necessary to assume pseudo-zero bits in specific locations. The quasi-octal representation is discussed in the D17B Computer Programming Manual (Ref 1:8). The same type of correlation was also used in designating a sector location in the FORTRAN program. Fig. 3 shows the correlation that was made between the F-loop, V-loop, R-loop, E-loop, H-loop and a sector location.

The concept used in writing the simulation program was to have the person using it provide the same data to the program as he would if he were using the actual computer in the laboratory. The switches must be turned to the proper positions to accomplish loading and computing. The data must be error free to successfully execute a program. The type of display (register or memory) is specified by the user.

The D17B computer simulation program consists of a main program and eight subroutines. The main program is a compilation of three distinct sections each of which performs a major function. These three sections are the reading and translation section, the noncompute mode section, and the compute mode section. Fig. 4 shows the program flow between these sections of the main program and the subroutines.

The organization and structure of each of the sections

SECTOR		V-LOOP R-LOOP F-LOOP		E-LOOP		H-LOOP	
MM	PORTMAN	MM	PORTMAN	MM	PORTMAN	MM	PORTMAN
000	1	0	1	0	1	0	1
001	2	1	2	1	2	1	2
002	3	2	3	2	3	2	3
003	4	3	4	3	4	3	4
004	5	0	1	4	5	4	5
005	6	1	2	5	6	5	6
006	7	2	3	6	7	6	7
007	8	3	4	7	8	7	8
010	9	0	1	0	1	10	9
011	10	1	2	1	2	11	10
012	11	2	3	2	3	12	11
013	12	3	4	3	4	13	12
014	13	0	1	4	5	14	13
015	14	1	2	5	6	15	14
016	15	2	3	6	7	16	15
017	16	3	4	7	8	17	16
020	17	0	1	0	1	0	1
021	18	1	2	1	2	1	2
022	19	2	3	2	3	2	3
023	20	3	4	3	4	3	4
024	21	0	1	4	5	4	5
.
.
170	121	0	1	0	1	10	9
171	122	1	2	1	2	11	10
172	123	2	3	2	3	12	11
173	124	3	4	3	4	13	12
174	125	0	1	4	5	14	13
175	126	1	2	5	6	15	14
176	127	2	3	6	7	16	15
177	128	3	4	7	8	17	16

Fig. 3. Correlation between MINIBOON and PORTMAN designation for Sector, Track and Multi-word Loops

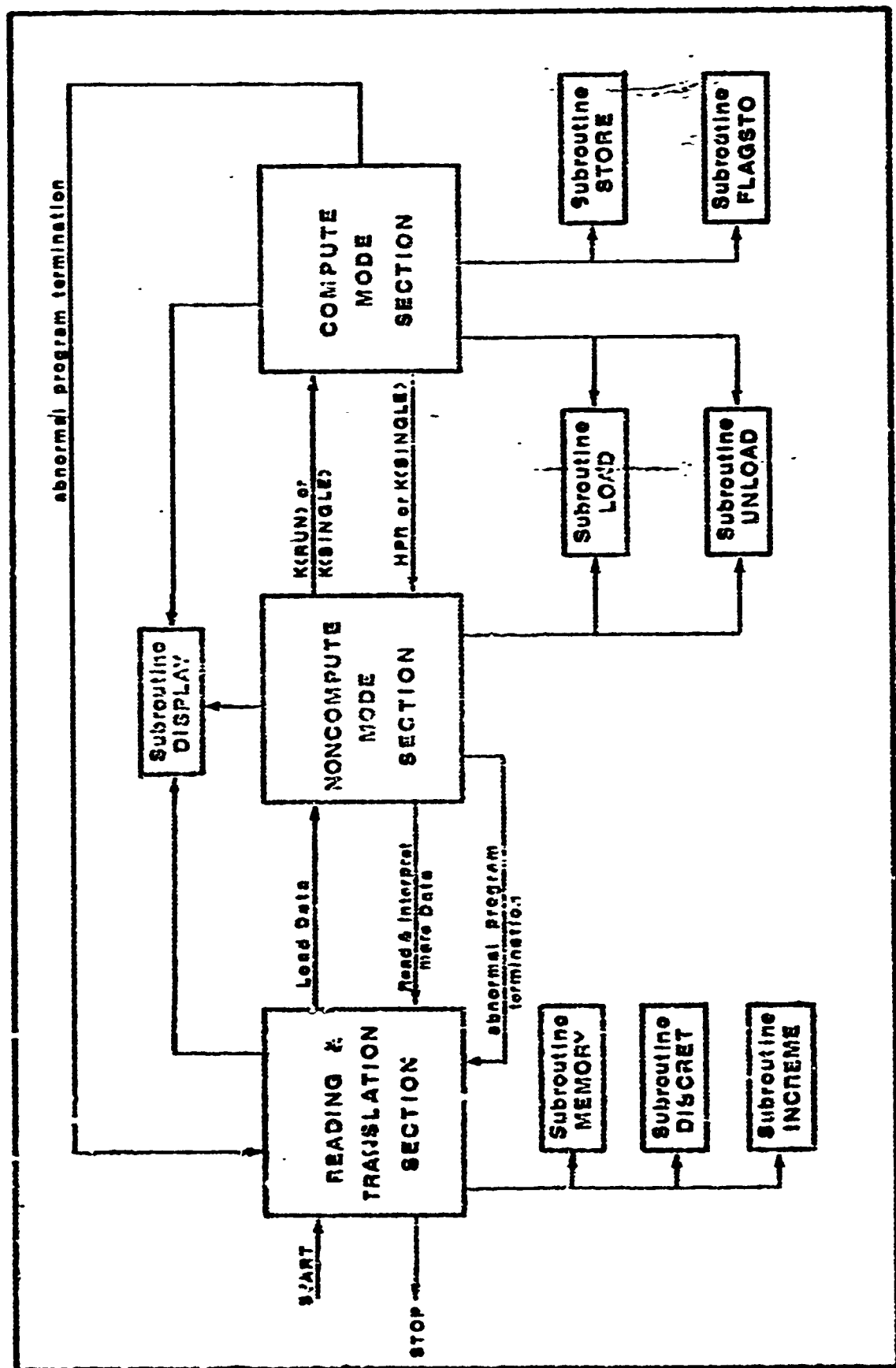


Fig. 4. 1972 Computer Simulation Program Flow

of the main program will be discussed along with the functions performed by each of the subroutines. The variables used in creating the simulation program are also listed with a short description of how each is used.

Reading and Translation Section. The reading and translation section is the translator and interpreter portion of the simulation program. All input data is read, interpreted, and translated in this portion of the main program. A transfer of operation to the noncompute node or one of the subroutines is made to utilize this data. The programming language accepted as valid data by the simulation program is described in detail in chapter III of this thesis and will not be discussed in the following description of the reading and translation section.

The reading and translation section is physically located at the beginning of the simulation program. When the simulation program is loaded for execution, execution begins at the start of this section. The first output produced by this section is a heading containing the name of the simulation program, the date, and the time at the beginning of execution. The remainder of the reading and translation section is responsible for the reading, interpreting, and translating of input data. Input data is read as alphabetic or numeric characters (hollerith). This data is then interpreted as octal or binary data, a load code, a switch designation (setting), or a miscellaneous input or command. The miscellaneous inputs or commands are

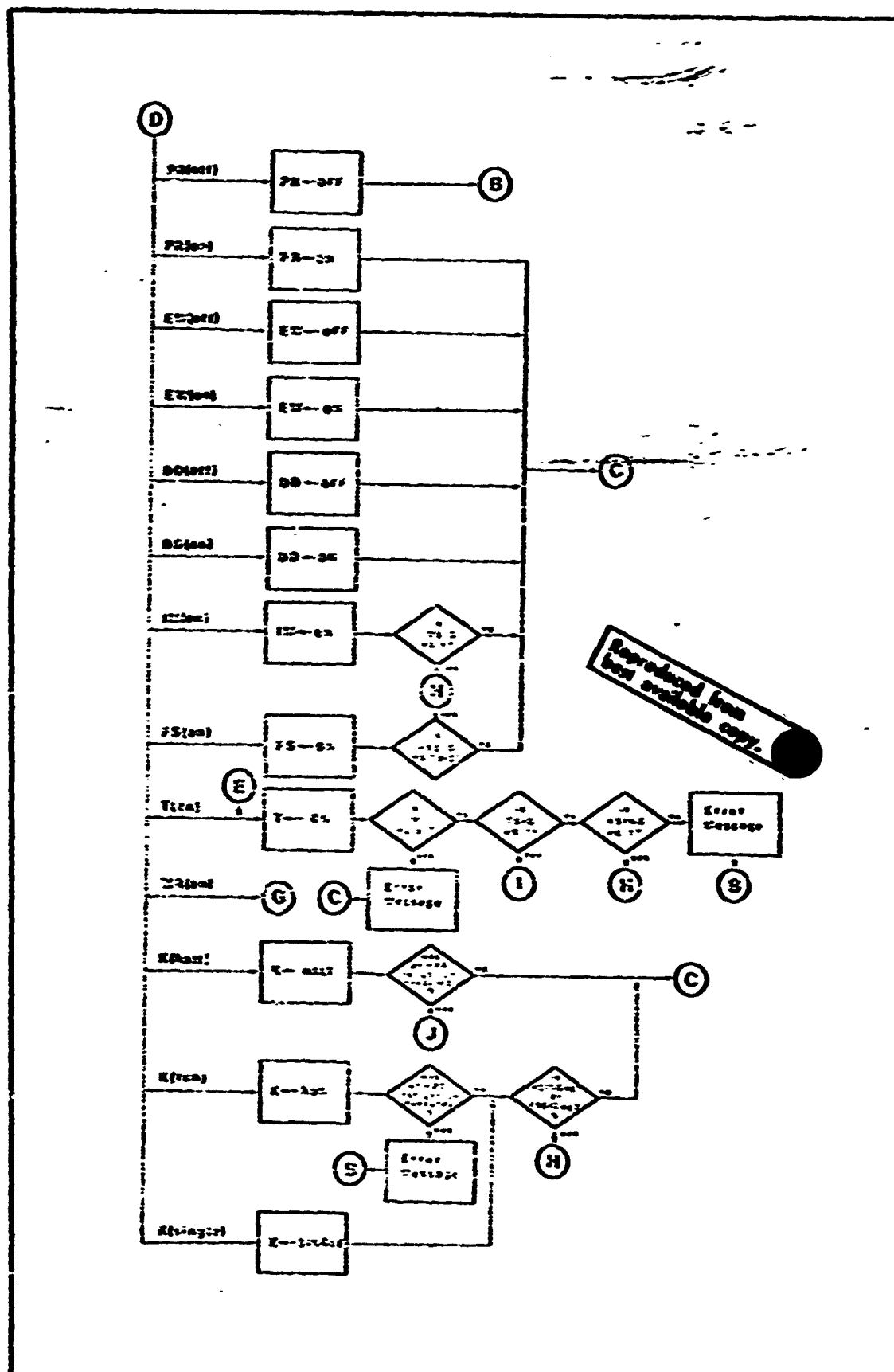


Fig. 6. Switch Interpretation Flowchart

responsible for a variety of functions which include the following: register and memory display, discrete data, incremental data, setting of flipflops, and mode tracing.

If data is interpreted as octal data, binary data, or a load code, a translation is made from the hollerith representation to binary integer data consisting of 1's and 0's. This binary integer data is then supplied to the noncompute mode of the simulation program where it is utilized. A switch designation results in a switch variable being loaded with the designation. Miscellaneous input or command data results in either the storing of input data or the flagging (setting to 1 or 0) of variables which control program flow.

Flow charts showing the organization and structure of the reading and translation section are shown in Figs. 5 thru 7. Fig. 5 shows the interaction and interpretative capability of the simulation program. Figs. 6 and 7 are extensions of the flowchart shown in Fig. 5 and show the results of interpretation of switch designations and interpretation of miscellaneous inputs and commands.

Non-Compute Mode Section. This section of the simulation program simulates the noncompute mode of the D17E computer. The noncompute mode can be divided into two categories each performing a major function. These two categories are noncompute load and noncompute nonload. The noncompute nonload function comprises the following modes: "prepare to operate", "sync bit counter 1", "sync bit counter 2", "manual halt",

and "program halt". In the prepare-to-operate mode, mode control flipflops are initialized. During sync bit counter 1 and 2 modes, a synchronizing between the clock track bit counter and the bits of a word is made. The manual halt mode is used for idling, preparing to load, and preparing to compute. The program halt mode is entered by execution of a program halt instruction. The noncompute load function is made up of the following modes: "wait", "prepare to sample", "sample code", "parity check", and "process code". The wait mode is used for idling while waiting for input data. The prepare to sample mode is entered when data is detected on the input lines. In the sample code mode the input data is read. The parity check mode is used for checking the input data for odd parity. In the process code mode the input data is decoded and processed according to the deciphered code.

Fig. 8 is a flowchart of the noncompute mode which shows the program flow between these modes. This flowchart was used in writing this part of the simulation program. Fig. 8 is drawn as a veitch diagram with the mode control flipflop states as the boolean variables. As an aid in tracing through the noncompute mode section of the simulation program, the mode control flipflops have been included as comment cards. The mode control flipflops are listed as being set to "1" or "0". In the D173 computer these flipflops were actually set, however, they were not needed in the simulation program because of the sequential flow

21

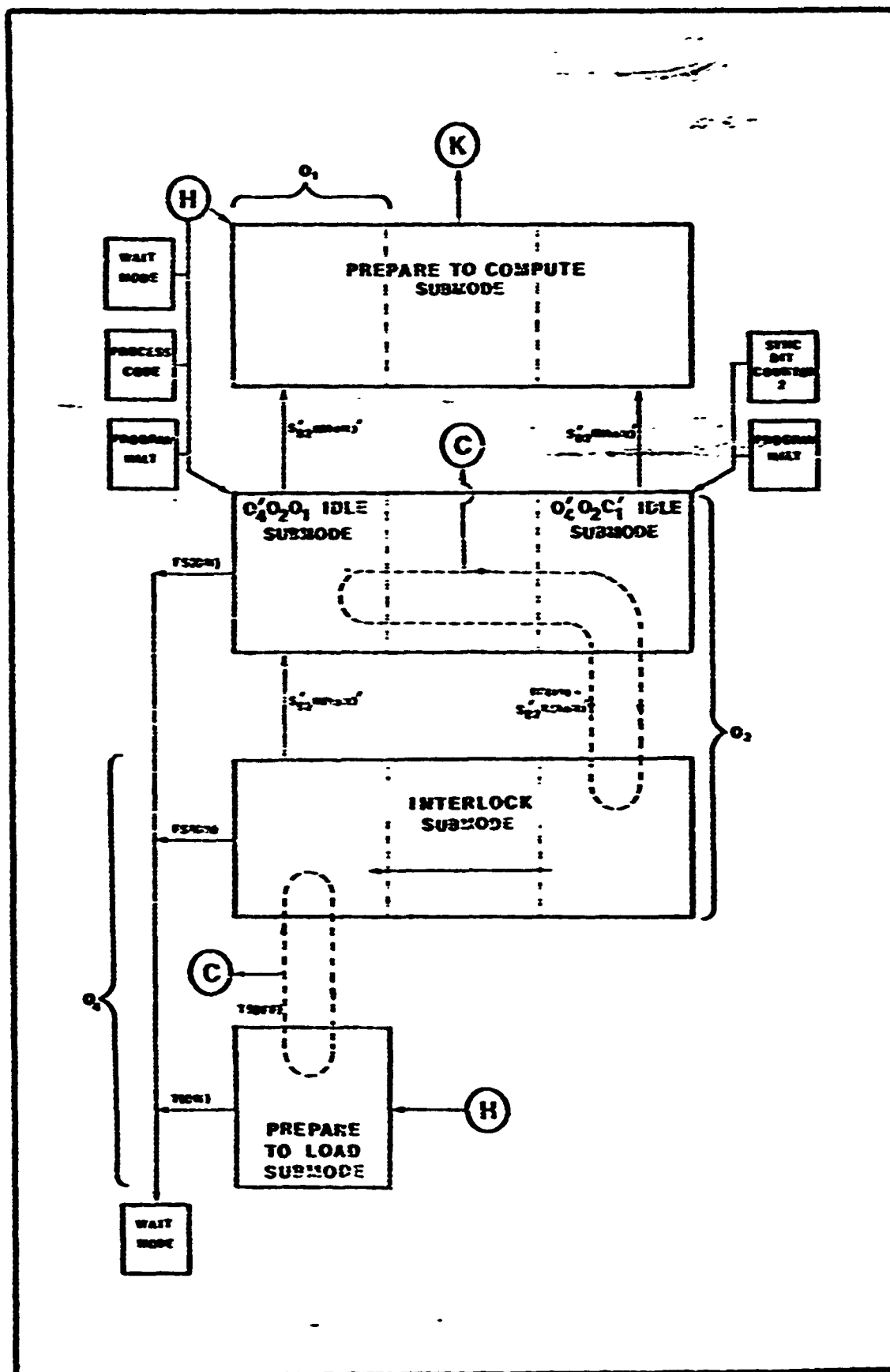


Fig. 9. Manual Halt Mode Flowchart

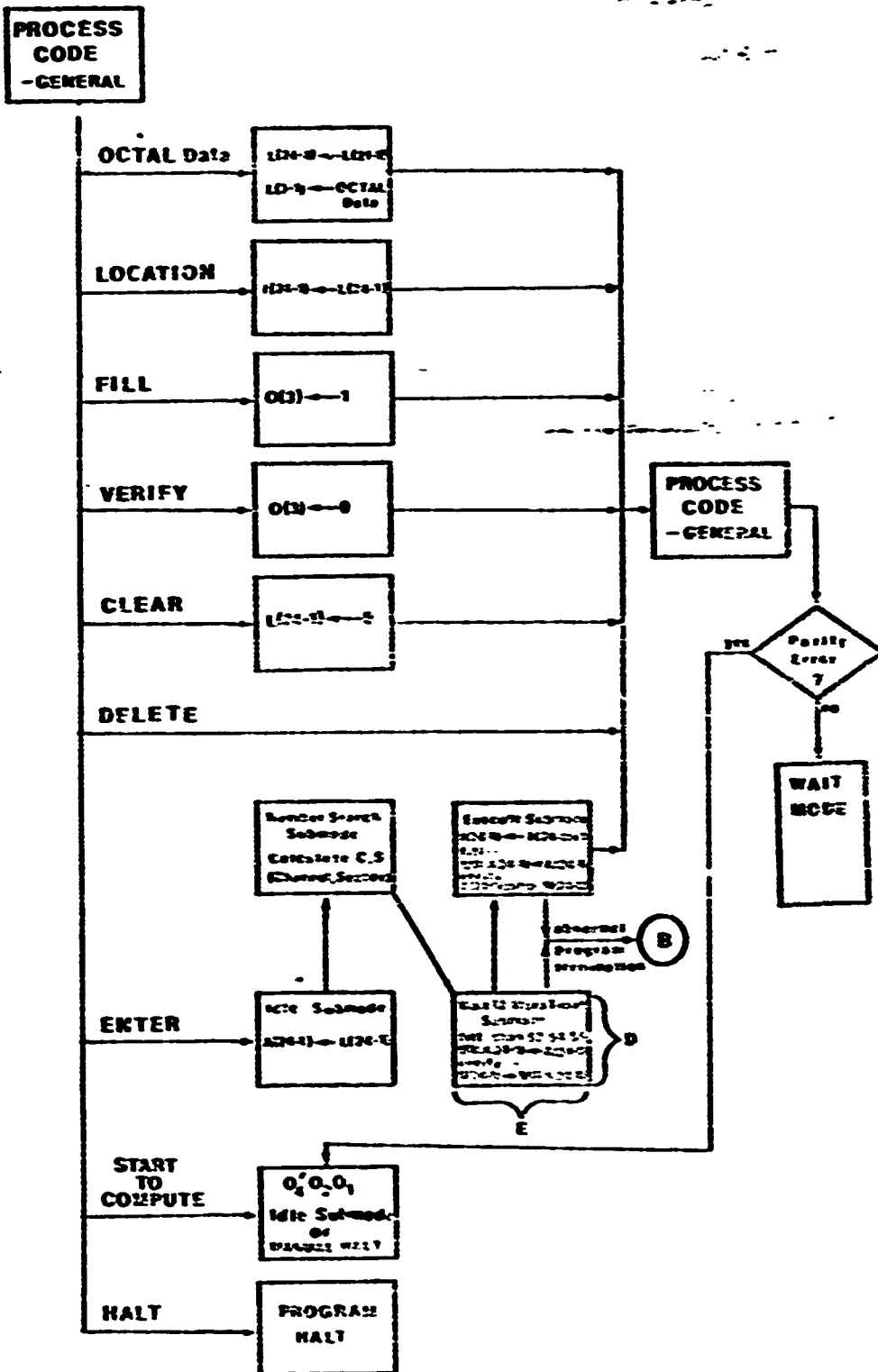


Fig. 10. Process Code Node Flowchart

that occurs in a FORTRAN program. Two additional figures, Figs. 9 and 10 have been included which give a further breakdown of the program flow in the "manual halt" mode of noncompute nonload and in the "process code" mode of noncompute load.

Compute Mode Section. This section of the simulation program simulates the compute mode of the D173 computer. The compute mode consists of modes which perform five major functions: "number search", "number read", "instruction search", "instruction read", and "execute". The number search, number read, instruction search, and instruction read modes are equivalent to the fetch cycle associated with other computers. The execute mode is equivalent to the execute cycle. Number search and instruction search locate the data word in memory while number read and instruction read unload the located word from memory into a register. Execute results in the execution of one of the 39 instructions in the instruction set of the D173 computer. The compute mode section of the simulation program was written using the above functions. A flowchart which shows the program flow in the compute mode section is given in Fig. 11.

Subroutines. The subroutines associated with the D173 computer simulation program were made for three purposes:

1. Those functions which were needed several times through the program were created as subroutines.

三

Subroutines falling into this category are Subroutine LOAD, Subroutine UNLOAD, and Subroutine DISPLAY.

2. Those functions which are only called from one place in the main program, but which are of such importance and magnitude that a separate location is beneficial in the organization of the simulation program. Subroutines in this category are Subroutine STORE, Subroutine FLAGSTO, and Subroutine MEMORY.

3. Those functions which will not be used very frequently. Therefore they could be removed from the simulation program if it was determined that they were not really needed. This would result in a decreased memory core size needed for execution of the simulation program. However, to be able to utilize all the instruction set of the D173 computer and all the channel designations these functions had to remain as a part of the simulation program. Subroutines in this category are Subroutine DISCRET and Subroutine INCREASE.

A description of the function performed by each of the subroutines will be given. The order of explanation is the order of appearance on the program listing located in Appendix A.

Subroutine STORE. This subroutine performs the store (STO) instruction, which stores the contents of the accumulator in the memory address given in bits 12 thru 1 of the instruction register. A normal loading of memory is performed for all channel designations except the single word channels (A, I, L, O), channel 70 (A-loop), and channel

72 (R-loop). The store instruction cannot store in single word loops. Storage in channels 70 and 72 provides the D17B with real time control. Storage in channel 70 results in a whole word addition of the accumulator contents to the addressed word of the R-loop if the fine countdown flipflop (FC) is "0" set. If FC is "1" set, a normal store takes place. Storage in channel 72 results in a split word addition of the accumulator contents to the addressed word of the R-loop if FC is "0" set. If FC is "1" set no store is allowed. This subroutine can detect erroneous switch settings which terminate the run upon return to the main program.

Subroutine LOAD. This subroutine provides the function of loading the contents of the accumulator into addressed memory locations. The areas of memory that can be loaded by the subroutine are: 20 cold-storage memory channels (channels 00 thru 45), the hot-storage memory channel (channel 50), channel 52 (R-loop), channel 54 (H-loop), channel 56 (E-loop), and channel 60 (U-loop). This subroutine can also detect erroneous switch settings which will terminate the program run. A call is made to this subroutine from both the noncompute mode section and the compute mode section of the main program.

Subroutine UNLOAD. This subroutine performs the function of unloading an addressed word of memory into the R-register. The information unloaded is then used either

as an instruction or an operand number. This subroutine can unload data from all addressable memory channels. Data unloaded from channels 70 and 72 is incremental data used for real time control. If the channel designation is either 70 (V-loop) or 72 (R-loop), one of two possible actions can take place. For the V-loop, if VK (incremental input flipflop) is "0" set then normal unloading occurs, however, if VK is "1" set then the one's complement of the V-loop is unloaded. The same conditions apply to the R-loop and the settings of RK (incremental input flipflop). This subroutine can detect out of range conditions for the cold-storage and hot-storage memory channels.

Subroutine FLAGSTORE. This subroutine performs the function of deciphering flag store location bits (bits 17, 18, 19) of the instruction register and storing the contents of the accumulator in the deciphered channel at the sector address associated with the execution of the present instruction. The flag store codes provide for storing in the following channels: hot-storage memory channel (channel 50), channel 52 (F-loop), channel 54 (H-loop), channel 56 (E-loop), channel 60 (R-loop), and channel 64 (L-register). The remaining two flag store possibilities are flag store telemetry signal and flag store idle.

Subroutine DISPLAY. Subroutine DISPLAY provides the simulation program with the capability of displaying the binary contents of all registers and loops. This subroutine

has two entry points, ENTRY REG1 and ENTRY REG2. Entry point REG1 is called from the reading and translation section of the main program and performs the function of interpreting the arguments given with the register command. An argument consists of a register designation enclosed in parenthesis immediately following a register command. If a valid argument exists, the variable Registr is set to one. Registr being one set allows the main program to denote a register or loop to be displayed. A call to entry point REG2 is then made to determine if the contents of that register or loop should be displayed. Entry point REG2 checks to see if the register or loop was specified in the register command argument, if not a return is made to the main program. If it was specified then its contents will be displayed as output.

Subroutine MEMORY. This subroutine provides the capability of displaying the contents of memory (channels 00 thru 50) whenever the memory command is specified. Upon entry into this subroutine a check is made of the memory command argument to determine if the display should be in octal or binary. A memory command argument consists of either BINARY or OCTAL being enclosed in parenthesis immediately following a memory command. If no argument was specified, the default condition of OCTAL is used. In displaying the contents of memory, only those portions of memory that have been written into since memory was last initialized will be shown in the output listing. Memory is initialized by writing ten decimal 9's into each word of

memory. This condition is then checked to determine if the contents have changed, and if they have, the contents of that location are printed as output.

Subroutine DISCRET. This subroutine provides the capability of entering discrete data and storing it for use in a program using the discrete input instructions (DIA or DIB). Subroutine DISCRET has two entry points, ENTRY DISK and ENTRY DIB. Entry point DISK is called to interpret and translate X-discrete inputs and store them for use during a program run. 19 bits make up each X-discrete input. A maximum of ten X-discrete input requests is allowed, because the storage area data array in the FORTRAN program is dimensioned for 10.

Entry point DIB is called to interpret and translate Y-discrete inputs and store. Each Y-discrete input consists of 24 bits. A maximum of ten Y-discrete input requests is allowed, because the storage area data array in the FORTRAN program is dimensioned for 10.

Subroutine INCRERE. This subroutine provides the capability for entering incremental data into the four words of the V-loop or the four words of the R-loop. Subroutine INCRERE has two entry points, ENTRY INCR and ENTRY INCR. Entry point INCR is called to interpret and translate R-incremental inputs and store in R-loop. Each R-incremental input is made up of 24 bits. Four R-incremental input requests fill the R-loop and entering another request

causes word 0 of the R-loop to be loaded with the new data. Additional inputs fill word 1, word 2, and word 3 with new data. This sequence can continue indefinitely.

Entry INCV is called to interpret and translate V-incremental inputs and store in V-loop. Each V-incremental input request is composed of 24 bits. Four V-incremental input requests fill the four words of the V-loop and additional input requests cycle through the four words again filling them with new data similar to the R-loop.

Simulation Program Variables. This section contains a compilation and description of the variables used in the D17B computer simulation program. A complete alphabetical listing is made of the variables with descriptions of their main uses within the program. Several of the variables have been used for more than one function. They are described as having no main usage. For these particular variables and several others, the reader should refer to the computer printout and note the use made of them in each instance they have been used. The variables that are in this category are discussed at the end of this section.

Each of the variables in the simulation program are integer variables or have been declared as such in an INTEGER statement, except the variables "Volts" and "Voltage" which are real variables.

The listing and description of the variables is as follows:

A(24) - Accumulator, consists of 24 bits.

AK	- Carry, borrow flipflop.
BINARY	- Binary=0 represents octal designation. Binary=1 represents binary designation.
C(5)	- Operand channel register, consists of 5 flipflops.
CB(5)	- Operand channel buffer register, consists of 5 flipflops, copies the 5 least significant bits of instruction register, for "Y" special instructions.
CHAN	- Contains FORTRAN channel designation.
CODE	- Used mainly for FORTRAN operation code designation.
COMMON(24,2)	- Common registers used for storage and manipulation of information in program.
CP(5)	- Instruction channel register, consists of 5 flipflops.
D(5)	- Discrete output register, consists of 5 flipflops.
DD	- Discrete switch.
DISPLAY	- Set to "P" when output will be disposed to high speed printer; set to "N" otherwise.
DZ	- Detector reset flipflop.
E(8)	- E-loop, consists of 8 words.
EM	- Cold-storage memory write switch.
F(4)	- F-loop, consists of 4 words.
FC	- Fine countdown flipflop.
FD	- Initiate loading switch.
FEED	- Flag store sector designator.
FEED	- Set in manual halt mode to FD or FE.

switches to manual halt mode.

- G(3) - Binary output register, consists of 3 flipflops.
- H(16) - H-loop, consists of 16 words.
- HALT - Data word containing hollerith characters "HALT".
- HCODE(8) - Data array which contains hexadecimal code for numbers 8 thru 15.
- I(24) - Instruction register, consists of 24 flipflops.
- ICHAN - Contains calculated ~~PORTAL~~ designation for instruction channel.
- IH - Mechanical input switch.
- INSIG - Set in manual halt mode to flag IH switch to manual halt mode.
- IRSE - Set to designate memory data to be unloaded into instruction register rather than number register.
- ISECT - Contains calculated ~~PORTAL~~ designation for instruction sector.
- IT(5) - Input transmission lines, contains the 4 bits of information and a parity bit which make up the input data for octal and load codes.
- K - Compute mode switch, three-position switch (Run, Halt, Single).
- KHPR - Set to 1 by an HPR instruction, used in program control after an HPR instruction has been executed.
- KSIG - Set to 1 by an HPR instruction, used to flag K switch to program halt mode.

- KSING - Set to 1 in manual halt mode to flag K switch or T signal to manual halt mode.
- L(24) - Lower accumulator, consists of 24 bits.
- LIST - Contains number of executions specified, has a default of 50.
- LIST1 - Counts the number of executions in compute mode and compares with the number specified.
- M(128,21) - Memory storage array, consists of 2688 words which includes cold-storage and hot-storage memory.
- MA(8) - Data array with ASCII-code designation for load codes.
- MAN - Set to 1 when input is to be in ASCII-code; set to 2 when ASCII-code is to be read from Tape2; set to 3 when ASCII-code is to be read from Tape3; reset to 0 at end of ASCII-code.
- MR - Master reset switch, (momentary on type).
- N(24) - Number register, consists of 24 bits.
- NBLANK - Data word containing hollerith character " ".
- NCHAN - Contains calculated FORTRAN designation for operand channel.
- NC.L - Counts input data columns, reset to 0 at count of 73.
- NC.MA - Data word containing hollerith character " , ".
- NP.LG - Set to 1 in a subprogram when fatal error has been encountered, terminates run upon return to main program.

NINES	- Data word containing 9999999999, used to initialize memory and detect out of range conditions.
NLIST	- Data array containing decimal integers 0 thru 9, used in detecting valid argument in execute command.
NLPAREN	- Data word containing hollerith character '('.
NLOOP(10)	- Data array containing hollerith characters of the registers and loops, used in displaying the contents of loops and registers.
NRPAREN	- Data word containing hollerith character ')'.
NSECT	- Contains calculated FORTRAN designation for operand sector.
NUM	- Used in several different applications throughout program.
NU(27)	- Data array used mainly for interpreting data, switch settings, and commands.
NWORD(72)	- An array used to store 72 characters of input data which are to be interpreted.
O(4)	- Operation-code storage register, consists of 4 flipflops.
OFF	- Data word containing hollerith characters "OFF".
ON	- Data word containing hollerith characters "ON".
ONE	- Data word containing octal one.
P(3)	- Phase register, consists of 3 flipflops.
PHASE	- Used in several different applications

through program.

- PR - Power on/off switch.
- R(4) - R-loop, incremental input loop consisting of 4 words.
- REG(10) - Array which is set by register command to display those registers and loops given as arguments.
- REGIST - Variable which contains the code of the register or loop that has just changed information.
- RI - Counts number of incremental inputs to R-loop, reset to 1 at count of 5.
- RII - R-loop incremental flipflop, can be set to 1 or 0 by program input.
- RUN - Data word containing hollerith characters "RUN".
- SB(3) - Flag code buffer register, consists of 3 flipflops, used in calculating location to which flagstore will take place.
- SECT - Used mainly to contain FORTRAN sector designation.
- SIGNAL - Variable set by signal command, if set to 1, modes of operation will be traced.
- SINGLE - Data word containing hollerith characters "SINGLE".
- T - Timing signal, (momentary on).
- TRAS - Set to 1 by TRA instruction, used in controlling program operation after a TRA instruction has been executed.
- TSIG - Set to 1 in wait mode, used to flag T signal to wait mode.

U	- U-loop, consists of 1 word.
V(4)	- V-loop, incremental input loop which consists of 4 words.
VI	- Counts number of incremental inputs to V-loop, reset to 1 at count of 5.
VR	- V-loop incremental flipflop, can be set to 1 or 0 by program input.
VO(8)	- Voltage output register, consists of 8 flipflops.
-VOLTAGE	- Variable used in calculating analog voltage designated by contents of voltage output register.
VOLTS	- Data array which contains numbers used in calculating voltage output.
WTIME	- Variable which is set to the number of word times required for execution of each instruction.
X(19,10)	- Array which stores X-discrete input data to be used in program run.
XI	- Counts number of X-discrete inputs, maximum of 10 is allowed.
X1	- Counts number of X-discrete inputs used in program, when greater than XI it assumes value of XI.
Y(24,10)	- Array which stores Y-discrete input data to be used in program run.
YI	- Counts number of Y-discrete inputs, maximum of 10 is allowed.
Y1	- Counts number of Y-discrete inputs used in program, when greater than YI it assumes value of YI.

ZERO

- Data word containing octal zero.

The major portion of the variables included in this listing have the same name as used in the D173 computer literature. For comparison purposes, the reader is referred to the documents pertaining to the D173 computer listed in the bibliography. (Ref 6:110-114)

The following variables, some of which appear in the above listing, have been used in several different applications in the simulation program: "Code", "Correg", "Rus", "Phase", "Sect", "Voltage", "I1", "I2", "I3", "I4", "I5", and "I6". These variables have been pointed out for those interested in modifying the simulation program or for those interested in implementing the simulation program on a different computer system.

A description of the organization and structure of the D173 computer simulation program has been given in this chapter. The next area to be covered is the simulation language accepted by the simulation program.

III. D173 Computer Simulation Language

This chapter describes the simulation language understood and accepted by the D173 computer simulation program. The simulation language is the input data to the simulation program. Methods for programming the simulated computer are discussed along with a method for creating a shortened version of the simulation language. Error detection capabilities of the simulation program are presented in chapter IV. Chapter V will present some examples of programs that have been run along with the types of output that are available.

For purposes of presentation, the simulation program language is divided into the following categories: numbers and load codes, switches, and miscellaneous inputs and commands. A description of the elements of the simulation language in each of these categories will be given along with guidelines for using each.

Numbers and Load Codes. The number systems and load codes accepted by the simulation program are:

Octal numbers - 0, 1, 2, 3, 4, 5, 6, 7

Binary numbers - 0, 1

Load Codes - HALT, LOCATION, FILL, VERIFY, CHECKSUM, ENTER, CLEAR, DELETE (A description of the Load Codes is given in Appendix B)

Three different representations of the numbers and load codes can be specified. By specifying OCTAL, BINARY, or HEX, an octal representation, a binary representation, or

an ASCII representation of the numbers and load codes can be used. The representation of the numbers and load codes in the three specifications are as follows:

	<u>Octal</u> <u>Representation</u>	<u>Binary</u> <u>Representation</u>	<u>ASCII</u> <u>Representation</u>
Numbers -			
	0	10000	0
	1	00001	1
	2	00010	2
	3	10011	3
	4	00100	4
	5	10101	5
	6	10110	6
	7	00111	7
Load Codes -			
HALT		01000	8
LOCATION		11001	9
FILL		11010	Z
VERIFY		01011	;
COMPUTE		11100	<
ENTER		01101	=
CLEAR		01110	^
DELETE		11111	?

When OCTAL is specified numbers and load codes must be in the octal representation. When BINARY is specified numbers and load codes must be in the binary representation. When ASCII is specified numbers and load codes must be in the ASCII representation. Program tapes to be run on the D17B

computer are in the ASCII representation. The default specification is OCTAL.

To terminate an octal or binary representation, all that is required is to specify another representation. To terminate an ASCII representation requires that the letter "R" be supplied after the last ASCII input symbol. Doing this will cause the program to revert to the octal representation or binary representation which it had before ~~ASCII~~ was specified.

The default specification is assumed if an error results in program termination, if the power switch is turned off, or if REINITIALIZATION is specified.

Switches. With the simulation language in this category it is possible to specify switches and designate a setting or mode. The simulation program accepts these switch designations and provides this information to program variables associated with the switches..

The form for specifying switches is as follows:

Switch(Arg)

where Switch is the designated switch mnemonic name, and Arg is the switch setting or mode position of the switch.

The switches and allowed settings are as follows:

<u>Switch Name</u>	<u>Switch Mnemonic & Settings</u>
Timing Signal	T(ON)
Power On/Off Switch	PR(ON), PR(OFF)
Initiate Loading Switch	PS(ON)

Master Reset Switch	MR(ON)
Cold-Storage Write Switch	EW(ON), EW(OFF)
Discrete Switch	DD(ON), DD(OFF)
Mechanical Input Switch	LI(ON)
Compute Mode Switch	K(HALT), K(SINGLE), K(RUN)

Timing Signal. The timing signal is produced automatically for the octal and ASCII representations. Therefore T(ON) need be used only after each binary representation of a number or load code. The timing signal is turned off by the program.

Power On/Off Switch. The power switch must be turned on after each loading of the binary deck, and this must be done before the master reset switch is turned on to prevent an abnormal program termination. Once the power switch is turned on, it remains on until it is turned off or the program is halted and "END OF PROGRAM" is printed. The default condition for the power switch is OFF.

Initiate Loading Switch. This switch is turned on to initiate loading and puts the simulation program in the wait mode of noncompute. It is a momentary on type switch and is turned off by the program.

Cold-Storage Write Switch. The cold-storage write switch is an on/off type switch that allows writing on the cold-storage channels (channels 00 thru 46) of memory when turned on. When the switch is off, no writing is allowed

and any attempt to write will cause an abnormal program termination. Once EX(OK) has been specified this condition will remain until it is turned off or until the program is halted and "END OF PROGRAM" is printed. The default condition for the cold-storage write switch is OFF.

Master Reset Switch. The master reset switch is turned on to initialize certain flipflops, synchronize the bit counter and sector track, load the instruction register with a transfer (TRA) instruction to channel 00, sector 000, and put the simulated computer into the manual halt mode of noncompute. The master reset switch is a momentary on type switch and is turned off by the program.

Discrete Switch. The discrete switch is an on/off type switch that allows writing on the hot-storage channel (channel 50) of memory and allows discrete outputs when turned on. When this switch is off no writing is allowed on channel 50 and any attempt to write will cause an abnormal program termination. Also when this switch is off no discrete outputs can appear which will be reflected in the output listing by the printing of the following statement: "DISCRETE SWITCH IS OFF - DISCRETE OUTPUTS ARE DISABLED". Once DD(OK) has been specified, this condition will remain until it is turned off or until the program is halted and "END OF PROGRAM" is printed. The default condition of the discrete switch is OFF.

Mechanical Input Switch. The mechanical input switch is used for putting the computer in the wait mode of non-compute from the idle submode of manual halt. Whenever IH(OH) is specified it must be followed by an ES(OH) to put the computer in the wait mode. Failure to do this causes an abnormal program termination. The mechanical input switch will be used very infrequently. The mechanical input switch is a momentary on type switch and is turned off by the program.

Compute Mode Switch. The compute mode switch is a three position switch that can be set at RUN, SINGLE, or HALT positions. With the switch set at the HALT position, only functions in the noncompute mode can be performed. Setting the switch at the RUN or SINGLE positions allows the computer to enter the compute mode. If the switch is set to the SINGLE position, one instruction is executed in the compute mode. The next instruction is stored in the instruction register and the simulated computer goes thru the program halt mode to the manual halt mode to wait for further switch settings or conditions. The SINGLE position of the compute mode switch is momentary on type and the program returns the switch to the HALT condition. When the compute mode switch is set to the RUN position, continuous operation occurs in the compute mode until an HIA (halt and proceed) instruction is encountered or the program is abnormally terminated. The compute mode switch has a default condition of HALT.

Miscellaneous Inputs and Commands. The simulation language in this category provides many functions that are unrelated but were not of such importance to warrant being in a category of their own. The functions that will be described in this category are listed as follows:

- Register and Memory Display
- Incremental Inputs
- Discrete Inputs
- Mode Tracing
- Execution Specifications
- Setting of Flipflops
- Initialization

Register and Memory Display. The binary contents of any of the registers (A,I,L,R) or loops (U,F,E,H,V,R) can be displayed by use of the register command. The register command has the following form:

REGISTER(Arg)

where Arg is a list of the registers and/or loops to be displayed.

The register command can contain from zero to ten specifications in the argument listing. A valid specification is one of the following letters which when specified will display the contents of the loop or register associated with it whenever the contents of that register or loop change in a program run:

<u>Specification</u>	<u>Register or Loop Displayed</u>
A	Accumulator
I	Instruction Register
L	Lower Accumulator
H	Number Register
U	U-loop (1-word loop)
F	F-loop (4-word loop)
E	E-loop (8-word loop)
H	H-loop (16-word loop)
V	V-loop (4-word input loop)
R	R-loop (4-word input loop)

The arguments of the register command can be separated by commas or blanks or they can be placed one after another.

Examples: REGISTER(A,I,L,H) Will display contents of accumulator, instruction register, lower accumulator, and number register.

REGISTER(V,R,A) V-loop, R-loop and accumulator will be displayed.

REGISTER() No registers or loops will be displayed.

The default condition for the register command is REGISTER(). The default condition is assumed each time a program run is terminated and more input data is supplied. Therefore if register display is wanted, a register command must be used each time data is entered.

To display the contents of cold-storage and hot-storage channels (channels 00 thru 50) of memory, a memory command is used. The memory command has the following form:

MEMORY(Arg)

where Arg is the type of display requested, either BINARY or OCTAL.

If the argument is not included or incorrectly specified, the default for Arg is OCTAL. The memory command can be used anywhere within the program.

Examples: MEMORY(BINARY) Memory display will be in binary.
MEMORY Memory display will be in octal,

Incremental Inputs. Because the simulation program does not have real time control capability, provisions were made for entering data in the V-loop and R-loop. This data could then be used in the programming of the simulated computer as though it had been supplied incrementally during real time processing.

The form of the request for entering incremental data is:

Loop(Arg)

where Loop is either V or R and Arg is 24 bits.

For Arg to fill the whole word of the V-loop or R-loop, it should contain 24 or more bits. Any bits above 24 are ignored. Any bits less than 24 results in the least significant bits remaining unchanged. Storage starts with bit 24 and continues towards bit 1 until data is exhausted.

Invalid bits are assumed by the program to be zero and a message is output to this effect. The first incremental input request stores data in word 0 of the V-loop or R-loop (whichever is being filled). The second request in

word 1, the third request in word 2, the fourth request in word 3. Additional requests start over with word 0 and repeat the cycle. Each time the program terminates for more data, the program is initialized to start with word 0 again. It is possible by including no data in the argument to skip numbers without changing the previously stored data. For readability blanks are ignored in the argument portion of the request. The data can therefore be arranged in any groupings desired.

Examples: R(000 001 010 011 100 101 110 111) This request fills word 0 of the R-loop with the binary data given in argument.

V() V() V() V(000000111111 000000 111111) This request causes words 0, 1, and 2 of the V-loop to remain unchanneled and word 3 to be filled with the binary data given in argument.

Discrete Inputs. Discrete inputs are necessary to supply data for use with the DIA and DIB (Discrete Input A and Discrete Input B) instructions.

The form of the request for entering discrete input is:

Type(Arg)

where Type is the type of discrete input, either X or Y, and Arg is 19 bits for X-discrete inputs and 24 bits for Y-discrete inputs.

For Arg to be valid it must contain the number of bits required for the input type. It can contain more bits than required, because excessive bits above those required are ignored. If not enough bits are supplied, however, the program will use the data beyond the request until the proper

count is reached.

Invalid bits are assumed by the program to be zero and a message is output to this effect. A maximum of ten X-discrete input requests and ten Y-discrete input requests is allowed before the storage area is filled. Additional requests are ignored. Input requests fill the storage array in sequential order from 1 to 10. Provisions for refilling the discrete storage array once all ten storage areas have been filled is given in the initialization portion of this section.

In using the stored discrete inputs, they are used from 1 to the highest number stored. Additional requests beyond the highest number results in the program using the highest number stored and a message is output to this effect. Whenever an abnormal termination of the program is made or the power switch is turned off, the counter for using discrete inputs is initialized to 1.

Code Tracing. Code tracing is used in deciphering the contents of a program. In the noncompute mode, the modes of operation are listed as output. In the compute mode, the instruction being executed is listed as output and a flag store is indicated if it was programmed.

The mode tracing capability is requested by a signal command with the following form:

SIGNAL

Whenever SIGNAL is specified in a program, it flips the representation of a variable from 1 to 0 or 0 to 1 depending

upon the value it had when the signal command was given. Code tracing is performed when the signal variable is 1. Whenever a program run is terminated and more input data is supplied, the signal variable is initialized to 0. The signal command used with the register command will give as output a detailed listing of the contents of a program.

Execution Specification. There are numerous occasions when a programmer will inadvertently write a program which loops on itself resulting in execution going on to infinity. To prevent this from happening in the simulated computer, provisions have been made for counting the number of execution cycles in the compute mode and terminating the program run when the number exceeds a specified amount. The programmer can specify the number of executions allowed by an execute command.

The form of the execute command is as follows:

EXECUTE(Arg)

where Arg is any four digit decimal number from 0000 to 9999.

If no execute command is given, the default value is EXECUTE(0050). Each time the program terminates for more data, the execution cycle counter is initialized to zero. However the number of executions allowed is initialized to the default value only upon an abnormal program termination, power switch turn off, or initialization. The number of executions specified is printed out whenever an execute command is encountered in a program.

Setting of Flipflops. The simulation language described here provides the capability of setting or resetting certain specified control flipflops which can change program flow when encountered. The flipflops that can be set are DR (detector) flipflop, RK (incremental input) flipflop, and FK (incremental input) flipflop.

The status of the DR flipflop is used in the execution of several instructions. It is reset to "0" by program control using the RSD (Reset Detector) instruction. To "1" set the DR flipflop, a command with the following form is used:

DR

The condition of the FK and RK flipflops determine the form of the data unloaded from the I-loop and J-loop respectively. If RK and FK are "0" set, data is unloaded into the R-register in normal form. If FK and RK are "1" set, the one's complement of the data is unloaded into the R-register.

The form for specifying the condition of the FK and RK flipflops is:

Flipflop(Arg)

where Flipflop is either FK or RK, and Arg is 0 or 1.

Initialization. When the binary deck of the simulation program is loaded for execution, memory is initialized by putting ten decimal 9's in every word location, the binary output flipflops are set to a +1 condition on all three lines, the discrete input request counters are set to start counting at 1, and the DR and RD flipflops are "0" set. The

programmer can cause the same initialization to occur by using the initialization command which has the following form:

REINITIALIZATION.

Programming Methods. The programming methods presented in this section and the programming examples of chapter 7 do not discuss methods for programming the D173 computer, but are concerned with methods and examples for programming the simulation program. For a discussion of programming the D173 computer, the reader should refer to the programming manual written for the Minuteman Computer Users Group, (Ref 1)

In the previous section of this chapter, a description of the input language that can be used by the simulation program was given. In this section methods will be described for arranging this language in a program form which can be run on the simulated computer.

The approach for arranging the input language in program form found most advantageous by the author is to visualize a hardware control console with switches for each element of the simulation language. To write a program then requires that the programmer write down the simulation language code for each switch that he would push on the console. This approach works because of the similarity between the simulation program and the hardware version of the computer.

In writing a program to be run on the simulated

computer, the programmer is not restricted to any input format. The input is format free and can be entered 72 characters per line. This allows the programmer to write a continuous program with each word of the simulation language separated by a delimiter. A delimiter is a character which fixes the end of a simulation language word. The delimiter required is one blank between each word of the simulation language. Exceptions to the use of a delimiter are that octal data can be grouped and no delimiter is needed for the ASCII representation.

There are three words in the simulation language that must begin in column one. These words are PR(OFF), GO, and \$. The two words PR(OFF) and GO signify that the input data is complete and ready to be read and interpreted by the simulation program. Entering PR(OFF) will result in the power switch being turned off at the end of the run. Entering GO causes the simulation program to return for more input data when it is interpreted. A comment line is created by specifying \$ in column 1. The program ignores the 71 remaining characters in that line.

Whenever the simulation program is loaded for execution, PR(ON) must be specified before PR(OFF). Once the power switch has been turned on it remains on until it is turned off or until the program is halted. SW(ON) and DD(ON) will also remain on until turned off or until the program is halted.

A typical program to be run on the simulated computer

will contain switch designations, octal or binary data and load codes, and commands. To enter octal or binary data and load codes, the simulated computer must be in the wait mode of noncompute. One of several ways that this can be accomplished is with the following switch designations:

PR(ON) RE(ON) PS(ON)

The power switch has now been turned on, the master reset switch has been depressed putting the computer through the "prepare to operate" mode, "sync bit counter 1-2-2" mode where the instruction register is loaded with a transfer (TRA) instruction to channel 00, sector 000, and into the manual halt mode. The initiate loading switch was pushed putting the computer into the wait mode. To enable writing on cold-storage channels of memory and allow the memory to be filled with input data the following would be specified:

EW(ON) FILL

The cold-storage write switch has been turned on and the fill load code has been specified. The simulated computer is now ready to receive and store data. The following program is a sample D17B computer program to add two octal numbers (14 and 2) and output a telemetry signal of the answer:

44010201 ENTER 64020202 ENTER 42402200 ENTER CLEAR 201
LOCATION CLEAR 14 ENTER CLEAR 2 ENTER

The addition program and data have been entered into memory. To put the simulated computer into the compute mode requires the following switch designations:

R(RUN) PR(ON)

The compute switch has been set at the RUN position and the master reset switch is depressed again putting the computer through the modes described earlier. Once the manual halt mode is reached, the computer will automatically go to the compute mode and the program will be executed.

The complete program would look as follows:

PR(ON) PR(ON) PS(ON) EX(ON) FILL 44010201 ENTER 64020202
ENTER 42102200 ENTER CLEAR 201 LOCATION CLEAR 14 ENTER
CLEAR 2 ENTER R(RUN) PR(ON)

The above program is complete and upon execution will output the answer (total 16) via a telemetry signal. The telemetry signal consists of 24 bits (000000 000000 000000 010000).

Entering either PR(OFF) or GO provides a ready signal to the simulation program. This ready signal results in the execution of the input data supplied by the user. If PR(OFF) was specified, then at the end of execution, the teletype would print the following message:

"TO RUN ENTER PR RUN TYPE 'RUN'; TO STOP TYPE 'HALT' - "

To continue running more programs or more data, the user would type RUN. The system would respond and tell the

user to (ENTER PROGRAM). At this point the user would enter more programs or more data.

If GO has been specified then at the completion of the program run the system would respond with the message to (CONTINUE PROGRAM). The user can now execute the same program over again with the same data or new data can be entered. The program written onto the memory will remain there until overwritten or until initialization occurs.

To execute the same program again would require the following:

HR(CH) K(RUN)

GO

To execute the same program again with new data (20 & 12) would require the following:

HR(CH) FS(CH) FILL CLEAR 201 LOCATION CLEAR 20 ENTER

CLEAR 12 ENTER K(RUN) HR(CH)

GO

Overwriting the previous program with a program to multiply two numbers (+.04000000 and +.30000000) could be accomplished as follows:

HR(CH) FS(CH) FILL CLEAR 1 LOCATION 24020202 ENTER

CLEAR 201 LOCATION 02000000 ENTER 14000000 ENTER

GO

This program would produce no results because it was not entered into the compute mode. To execute it would

require:

```
RR(ON) R(RUN)
```

```
GO
```

Execution of the program produces a telemetry signal of the answer. For a more comprehensive listing of the compute mode portion of the program, the signal and register commands can be used. When using the signal and register commands in the compute mode for the first few times, the number of executions allowed should be lowered. This is done to prevent large amounts of output in case the program has loops. To execute the multiply program using the signal and register commands would require:

```
EXECUTE(0010) SIGNAL REGISTER(A,I,L,N) R(RUN) RR(ON)
```

```
GO
```

It is possible to run a program ending with an HPR instruction (such as the previous addition or multiply programs) several times using two different numbers each time. To do this requires specifying R(HALT) when the simulated computer is in the program halt mode and to follow this by RS(ON). Specifying R(HALT) puts the simulated computer in the manual halt mode and specifying RS(ON) puts the simulated computer in the wait mode. If a signal or register command was being used, then SIGNAL REGISTER() should be specified before entering more data. Doing this prevents mode tracing and register display while the new data is

being loaded into the computer. The signal and register commands must be respecified before K(RUN) is specified to allow mode tracing and register display in the compute mode. The new data can be entered by specifying a fill load code and following this with the new data. The simulation program can now be put back into the compute mode by specifying K(RUN) followed by IR(ON). This cycle can be repeated as many times as desired. The program for doing this with two sets of data would look as follows:

```
IR(ON) PS(ON) FILL CLEAR 201 LOCATION 20200000 ENTER
04040000 ENTER K(RUN) IR(ON) K(HALT) PS(ON) FILL
CLEAR 201 LOCATION 00300000 ENTER 22300000 ENTER
K(RUN) IR(ON)
GO
```

The previous examples have been entered into the compute mode by specifying IR(ON) K(RUN) or K(RUN) IR(ON). This results in a transfer to channel 00, sector 000 and starting execution at that location. A program can be executed at any starting location by specifying 5000xxxx LOCATION COMPUTE K(RUN). xxxx is the channel address and sector location of the first instruction to be executed. LOCATION puts the transfer instruction 5000xxxx in the instruction register, COMPUTE puts the computer in the manual halt mode and K(RUN) puts the computer in the compute mode for continuous run.

All programs which specify K(RUN) should end with a halt and proceed (H&P) instruction. This instruction puts

the computer into the program halt mode. If an HPR instruction is not used, the program will continue in the compute mode until an error occurs which will terminate the run or the number of executions exceeds the number specified.

Shortened Version of Simulation Language. The following description of a method for creating shortened versions of the simulation language does not apply to octal data, switches (except compute mode switch), or flipflop settings. These parts of the language have not been included in the discussion.

A shortened version of the input language can be created by the user. To do this requires taking those letters of a simulation language word which are used by the simulation program in interpreting it and using those letters as the input for the word. Additional letters can be added to build a mnemonic form of the word if desired. The following is a listing of those words in the simulation language which can be shortened, a listing of the portions of the word which are used in interpreting it, and an example of a shortened version of the word:

Simulation Language words which can be <u>Shortened</u>	Interpreting <u>Letters</u>	Example of a <u>Shortened Version</u>
HALT	H	HALT
LOCATION	L	LOC
FILL	FI	FILL
VERIFY	V	VER

COMPUTE	CO	CON
ENTER	EN	EN
CLEAR	CL	CL
DELETE	DE	DEL
OCTAL	O	OCT
BINARY	B	BIN
MEAN	MH	MEAN
SINGLE	S	SING
- RUN	- R	RUN
REGISTER(Arg)	RE(Arg)	REG(Arg)
MEMORY(BINARY)	ME(B)	MEM(B)
MEMORY(OCTAL)	ME(O)	MEM(O)
SIGNAL	S	SIG
EXECUTE(Arg)	EX(Arg)	EXEC(Arg)
REINITIALIZATION	REI	REINIT

The same using conditions apply to the abbreviated version of the language as apply to the full-word version. A blank is needed as a delimiter between each word of the language except octal data. Input data is entered with a free format. All 72 columns can be used for entering input data, however, no language element can be divided between two lines. If a word will not fit on a line, leave the remainder of the line blank and put the word at the first of the next line.

A description of the simulation has been given in this chapter. This language consists of numbers and load codes, switches, and miscellaneous commands. Also discussed was

the method for creating programs to be executed by the simulation program. The chapter was concluded with a method for creating a shortened version of the simulation language.

The following chapters will give a listing of the error detection capability of the simulation program and will present examples of programs that have been run on the simulated computer.

IV. Error Detection

Error detection is one of the outstanding features of the D17B computer simulation program. With this capability the program tapes can be error checked by the simulated computer before they are run on the D17B computer. To successfully load and execute a program on the D17B computer the program has to be error free. At the present time there are no error checks made by the D17B computer except for parity and verifying the contents of memory.

The error detection provided by the simulation program goes beyond checking just program tapes. All input data is checked for validity by comparing the input symbols against the simulation language symbols. Checks are also made by the simulation program to detect invalid switch settings, addresses that are out of range of the program, and a variety of conditions that are not allowed by the D17B computer.

A listing of the error statements that are provided by the simulation program is given in this section. Included with each statement are possible causes of the error or a further explanation of the error.

Error Statements/Causes of Errors. A listing of the error statements provided by the D17B computer simulation program is as follows:

THE FOLLOWING DATA IS NOT ALLOWED: (Invalid Data)
Input symbols specified are not part of the simulation language.

LOAD CODES MUST BE IN BINARY WHEN BINARY IS SPECIFIED

A different representation for a load code was used when the binary representation was specified.

THE FOLLOWING INPUT DATA IS INVALID: (Invalid Data)

Portions of the input word were interpreted but an improper symbol was encountered disallowing any further interpretation.

COMPUTER IS NOT IN WAIT MODE - DATA CANNOT BE ENTERED -
PROGRAM TERMINATED

The program must be in the wait mode of noncompute for data to be entered.

AN PS(ON) SIGNAL MUST FOLLOW AN IK(ON) SIGNAL TO PUT MACHINE
IN WAIT MODE - DATA IGNORED

If IK(ON) is specified, the only way to put the computer in the wait mode is to specify PS(ON).

COMPUTE MODE SWITCH SPECIFIED INCORRECTLY

Only RUN, SINGLE, or HALT can be used as modes for the compute switch.

THE FOLLOWING INPUT DATA ON PAPER TAPE IS INVALID - (Invalid
Symbol)

When PAPER has been specified, input data must be in the ASCII representation.

EXECUTE ARGUMENT SPECIFIED INCORRECTLY - DEFAULT VALUE OF 50
ASSUMED

The argument of the execute command must contain four decimal digits to be valid.

COLD-STORAGE WRITE SWITCH SPECIFIED INCORRECTLY

Only ON or OFF can be used as the settings for the cold-storage memory write switch.

DISCRETE SWITCH SPECIFIED INCORRECTLY

Only ON or OFF can be used as the settings for the discrete switch.

POWER ON/OFF DATA IS INCORRECT

Only ON or OFF can be used as the settings for the power on/off switch.

CHANNEL SPECIFIED CANNOT BE LOADED - PROGRAM TERMINATED

Only channels 00 thru 56 can be loaded by an enter load code.

PR(OFF) CANNOT BE SPECIFIED WITH PR(ON) - PROGRAM TERMINATED
PR(ON) must be specified before PR(OFF).

K(HALT) MUST BE SPECIFIED BEFORE K(RUN) AFTER AN HPR INSTRUCTION - PROGRAM TERMINATED

An HPR instruction puts the computer into the program halt mode and K(HALT) must be specified to get out of the program halt mode and into the manual halt mode.

A TRANSFER IS NOT ALLOWED TO L-REG, V-, OR R-LOOPS - PROGRAM TERMINATED

Channels 64, 70, and 72 cannot be used with a TRA (Transfer) instruction).

THE X-INSTRUCTION REQUESTED IS NOT AN INSTRUCTION

Not all possibilities for X-special instructions have been wired into the computer, you have specified one of these areas for execution. Check the Operation Code veitch diagram.

STORAGE CANNOT TAKE PLACE IN COLD-STORAGE CHANNELS IF COLD STORAGE WRITE SWITCH IS OFF - PROGRAM TERMINATED

EW(OFF) must be specified before writing can take place on cold-storage memory channels.

STORAGE IS NOT ALLOWED IN SINGLE-LOOPS (A,I,L, OR U) - PROGRAM TERMINATED

The STO (Store) instruction cannot store in single-word loops.

STORAGE CANNOT TAKE PLACE IN CHANNEL 50 IF DISCRETE SWITCH IS OFF - PROGRAM TERMINATED

DD(OFF) must be specified before writing can take place on

channel 50 (hot-storage memory channel).

COLD-STORAGE MEMORY CANNOT BE LOADED IF COLD-STORAGE WRITE SWITCH IS OFF - PROGRAM TERMINATED

EW(ON) must be specified before writing can take place on cold-storage memory channels.

HOT-STORAGE MEMORY CANNOT BE LOADED IF DISCRETE SWITCH IS OFF - PROGRAM TERMINATED

DD(CH) must be specified before writing can take place on channel 50 (hot-storage memory channel).

FLAGSTORE IS ALLOWED ONLY IN L-REG WHEN STORE INSTRUCTION IS TO CHAN 50, F, H, OR E-LOOPS - PROGRAM TERMINATED

If a STO (Store) instruction is to channels 50, 52, 54, or 56, a flag store is allowed only to channel 64.

OPERAND ADDRESS IS OUT OF RANGE - PROGRAM TERMINATED

An area of memory has been designated for an operand, but no data has been loaded there.

INSTRUCTION ADDRESS IS OUT OF RANGE - PROGRAM TERMINATED

A memory word has been designated as the next instruction, however no data has been loaded or stored into that memory word.

REGISTER DISPLAY REQUEST IS INVALID - (Invalid Register Display Request)

Register display command is missing a left parenthesis.

(Invalid Symbol) IS NOT A VALID REGISTER DISPLAY ARGUMENT

A symbol other than one of the ten register display symbols was used.

V. Programming Examples

This chapter is concerned with programs that have been run on the D17B computer simulation program. Seven different example programs and flow charts will be presented which show the types of output that can be realized. In the course of entering these seven examples, the majority of the instructions in the instruction set of the D17B computer will be used.

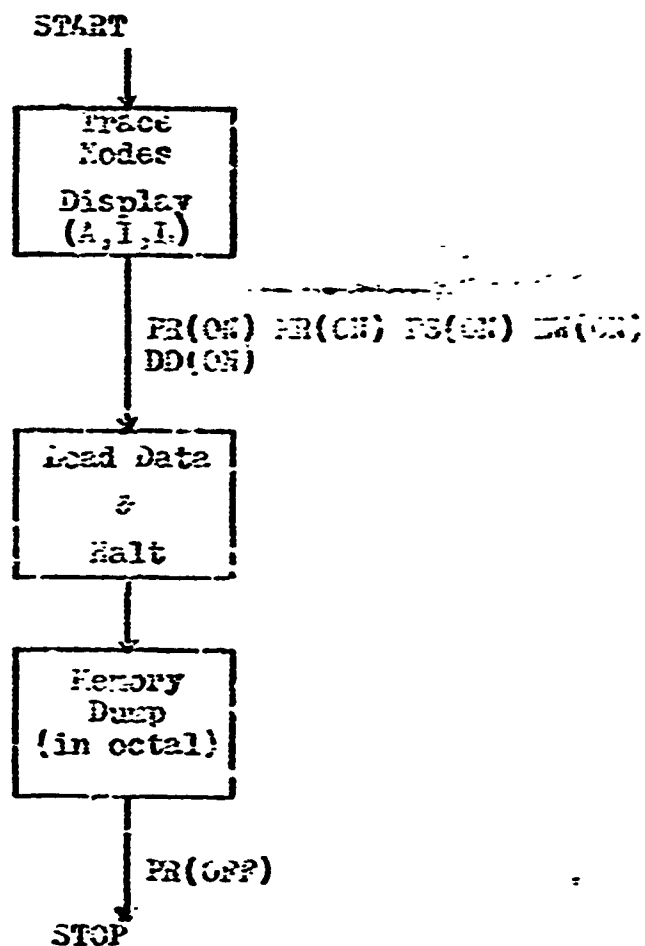
Example Program Number 1. In this example, the type of output available in the noncompute mode is shown. This example shows the way load codes are used and the way in which data is loaded into memory.

Signal tracing and register display in the noncompute mode will be used very infrequently because of the large amounts of output even for a small program. However, for someone learning the operation of the D17B computer the output displayed by these two commands can be used as a teaching aid.

A memory display request is made at the end of this example. The memory dump that appears on the listing has a channel and sector designation on the left. This address is the memory address of the octal word appearing in the first column of that row. The second column in the same row contains the memory word of the next sector location. The same applies to all remaining words in that row. If a sector location in a row has not been loaded with data it

appears on the listing as a word containing all 7's. The rows in which none of the sector locations are loaded with data do not appear in the memory dump listing.

The flowchart for example program no. 1 is as follows:



IF OUTPUT IS TO BE DISPOSED TO PRINTER, TYPE "P" AND "YOUR NAME"; OTHERWISE TYPE " " - "

```

*****
**                                     **
**                               D173 COMPUTER                               **
**                               SIMULATION PROGRAM                          **
**                                     **
**      DATE= 01/24/72              TIME= 19.43.00                          **
**                                     **
*****

```

** PRINTOUT OF INPUT PROGRAM **

..(ENTER PROGRAM)

SIGNAL REGISTER(A,1,L) PROCESS REGISTER(R,0,0) ENCODE ENCODE FILL
 CLEAR 01234567 ENTER HALT MEMORY(LOCAL)
 PROCESS

** RESULTS OF SIMULATION **

SIGNAL 02 - NODES WILL BE TRACED

POWER HAS BEEN TURNED ON

MASTER RESET SEQUENCE
 PREPARE TO OPERATE NODE
 SYNC BIT COUNTER 1 NODE
 SYNC BIT COUNTER 2 NODE
 LOG(4-1) = 101 000 000 000 000 000 000

0(4) 7(2)0(1) 1010 SUB-NODE OF MANUAL HALT
 INTERLOCK SUB-NODE OF MANUAL HALT
 PREPARE TO LOG SUB-NODE OF MANUAL HALT
 CIRCULATES BETWEEN PREPARE TO LOG AND INTERLOCK SUB-NODE OF MANUAL
 HALT
 WAIT NODE

PREPARE TO SAMPLE NODE
 SAMPLE CODE NODE
 PARITY CHECK NODE
 PROCESS CODE - FILL
 WAIT NODE

PREPARE TO SAMPLE NODE
 SAMPLE CODE NODE
 PARITY CHECK NODE
 PROCESS CODE - CLEAR
 LOG(4-1) = 000 000 000 000 000 000 000
 WAIT NODE

PREPARE TO SAMPLE NODE
 SAMPLE CODE NODE
 PARITY CHECK NODE
 PROCESS CODE - LOCAL
 LOG(4-1) = 100 000 000 000 000 000 000
 WAIT NODE

PREPARE TO SAMPLE MODE
 SAMPLE CODE MODE
 PARITY CHECK MODE
 PROCESS CODE - TOTAL
 L(24-1) = 000 000 000 000 000 000 000 001
 WAIT MODE

PREPARE TO SAMPLE MODE
 SAMPLE CODE MODE
 PARITY CHECK MODE
 PROCESS CODE - TOTAL
 L(24-1) = 000 000 000 000 000 000 001 010
 WAIT MODE

PREPARE TO SAMPLE MODE
 SAMPLE CODE MODE
 PARITY CHECK MODE
 PROCESS CODE - TOTAL
 L(24-1) = 000 000 000 000 000 001 010 011
 WAIT MODE

PREPARE TO SAMPLE MODE
 SAMPLE CODE MODE
 PARITY CHECK MODE
 PROCESS CODE - TOTAL
 L(24-1) = 000 000 000 000 001 010 011 100
 WAIT MODE

PREPARE TO SAMPLE MODE
 SAMPLE CODE MODE
 PARITY CHECK MODE
 PROCESS CODE - TOTAL
 L(24-1) = 000 000 000 001 010 011 100 101
 WAIT MODE

PREPARE TO SAMPLE MODE
 SAMPLE CODE MODE
 PARITY CHECK MODE
 PROCESS CODE - TOTAL
 L(24-1) = 000 000 001 010 101 100 101 110
 WAIT MODE

PREPARE TO SAMPLE MODE
 SAMPLE CODE MODE
 PARITY CHECK MODE
 PROCESS CODE - TOTAL
 L(24-1) = 000 001 010 101 100 101 110 111
 WAIT MODE

PREPARE TO SAMPLE MODE
 SAMPLE CODE MODE
 PARITY CHECK MODE
 PROCESS CODE - ENTER
 ENTER - TONE SUB-MODE OF FULL-ENTRY
 L(24-1) = 000 001 010 101 101 101 111 111
 ENTER - TONE SUB-MODE OF FULL-ENTRY
 L(24-1) = 001 001 010 101 101 101 111 111
 ENTER - TONE SUB-MODE OF FULL-ENTRY
 L(24-1) = 111 000 001 100 100 100 000 001
 WAIT MODE

GE/EE/72-7

PREPARE TO SAMPLE CODE
SAMPLE CODE MODE
PARITY CHECK CODE
PROCESS CODE - HALT
PROGRAM HALT MODE

S(4) P(2) S(1) - HOLD SUB-MODE OF MANUAL HALT
INTERLOCK SUB-MODE OF MANUAL HALT
PREPARE TO LOAD SUB-MODE OF MANUAL HALT
CIRCULATES BETWEEN PREPARE TO LOAD AND INTERLOCK SUB-MODES OF MANUAL
HALT

** MEMORY DUMP **

CHAS SECT
00 000 31234567 77777777 77777777 77777777

** END OF MEMORY DUMP ** (PORTIONS OF MEMORY NOT LISTED CONTAIN NO
INFORMATION PROVIDED BY THE PRESENT PROGRAM BUS)

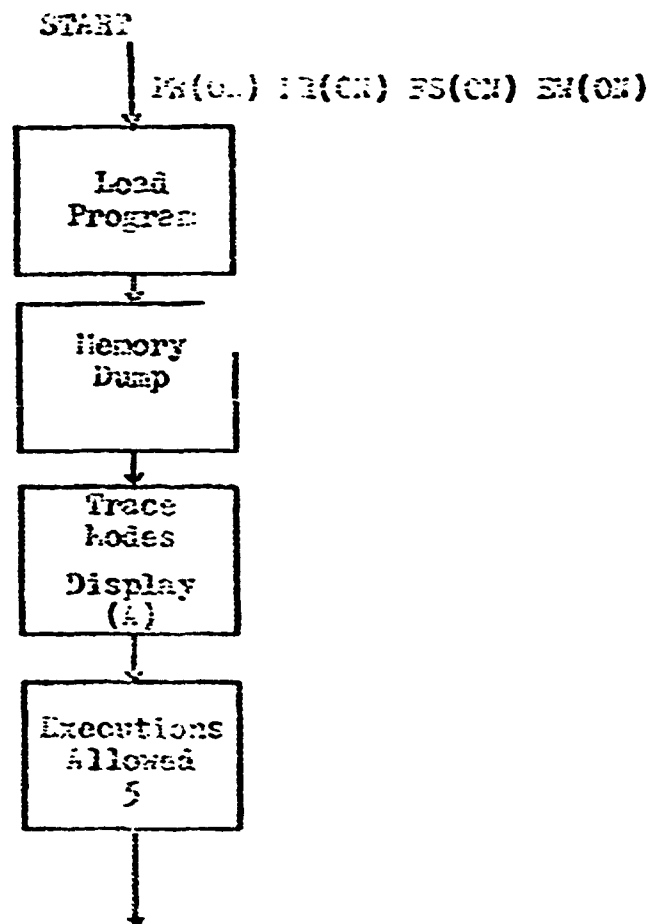
POWER HAS BEEN TURNED OFF
TO RUN ANOTHER PROGRAM: TYPE "BUS"; TO STOP TYPE "HALT" - HALT

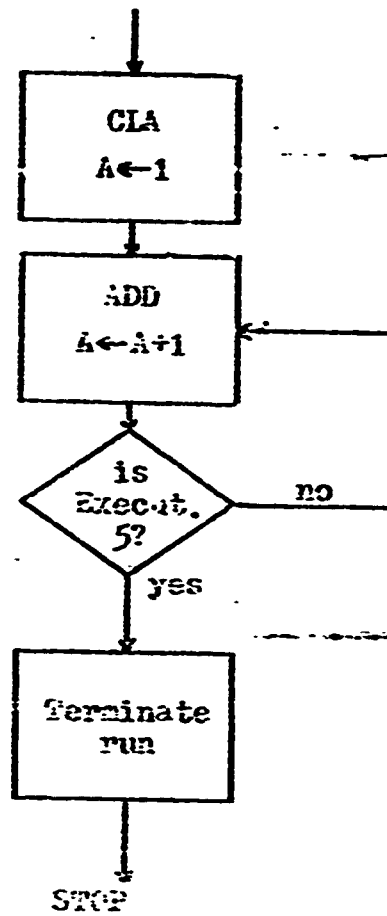
** END OF PROGRAM ** EXECUTION TIME: .050 SEC
19.01.07.070

Example Program Number 2. This example consists of an addition ripple program that has been written using the three different representations for input data. Each program has been loaded and executed separately. The addition ripple program loads 1 into the accumulator and keeps adding 1 to the contents of the accumulator. Because this program loops on itself, the execute command has been used to stop the executions at a count of five.

The initialization and memory commands are used with each representation. Code tracing and register display are used in the compute mode portion only.

The flowchart for example program no. 2 is as follows:





IF OUTPUT IS TO BE DISPOSED TO PRINTER, TYPE "P" AND "YOUR NAME"; OTHERWISE TYPE "N" - 2

```

*****
**                                     **
**                               5175 COMPUTER                               **
**                               SIMULATION PROGRAM                           **
**                                     **
**    DATE: 01/24/72                TIME: 19.54.45                          **
**                                     **
*****

```

** PRINTOUT OF INPUT PROGRAM **

(ENTER PROGRAM)

5 ADDITION RIFFLE PROGRAM IS TOTAL REPRESENTATION

PR(05) 05(05) 05(05) 05(05) 05(05) 05(05) 05(05) 05(05) 05(05) 05(05)
 44010002 ENTER 54010002 ENTER CLEAR 1 ENTER
 MEMORY SIGNAL REGISTER(01) EXECUTE(0005) 1(0005) 05(05)
 PR(05)

** RESULTS OF SIMULATION **

** MEMORY DUMP **

ADDR	SECT	DATA	DATA	DATA	DATA
00	000	44010002	54010002	00050000	77777777

** END OF MEMORY DUMP **

CONTENTS OF MEMORY NOT LISTED CONTAIN NO INFORMATION PRODUCED BY THE PRESENT PROGRAM (000)

SIGNAL 00 - MODES WILL BE TRACED

NO. OF EXECUTIONS SPECIFIED : 5
 MASTER FIRST PERCUSSION
 PREPARE TO OPERATE MODE
 STOP BIT COUNTER 1 MODE
 STOP BIT COUNTER 2 MODE

0(0) 0(0) 0(0) 0(0) 0(0) 0(0) 0(0) 0(0) 0(0) 0(0)
 PREPARE TO COMPUTE 0(0) MODE OF SIGNAL HALT

COMPUTE MODE

TRANSFER INSTRUCTION - (TRX)

CLEAR 1 0(0) INSTRUCTION - (CLR)
 A(24-1) : 000 000 000 000 000 000 000 000

ADD INSTRUCTION - (ADD)
 A(24-1) : 000 000 000 000 000 000 000 000

ADD INSTRUCTION - (ADD)
A(24-1) = 000 000 000 000 000 000 000 011

NO. OF EXECUTIONS HAVE EXCEEDED NO. SPECIFIED - PROGRAM TERMINATED
TO RUN ANOTHER PROGRAM TYPE "RUN"; TO STOP TYPE "HALT" - RUN

*(ENTER PROGRAM) ** PRINTOUT OF INPUT PROGRAM **

REINITIALIZE
GO

*(ENTER PROGRAM) ** RESULTS OF SIMULATION **

MEMORY HAS BEEN INITIALIZED

*(ENTER PROGRAM) ** PRINTOUT OF INPUT PROGRAM **

5 ADDITION RIFFLE PROGRAM IN BINARY REPRESENTATION

PROGRAM (HEX) PS(HEX) EX(HEX) BINARY
11010 T(HEX) 00100 T(HEX) 00100 T(HEX) 10000 T(HEX) 00001 T(HEX) 10000 T(HEX)
10000 T(HEX) 10000 T(HEX) 00010 T(HEX) 01101 T(HEX) 10110 T(HEX) 00100 T(HEX)
10000 T(HEX) 00001 T(HEX) 10000 T(HEX) 10000 T(HEX) 10000 T(HEX) 10000 T(HEX)
01101 T(HEX) 01110 T(HEX) 00001 T(HEX) 01101 T(HEX)
MEMORY SIGNAL REGISTER(A) EXECUTE(00000) EX(HEX) EX(HEX)
PR(OFF)

*(ENTER PROGRAM) ** RESULTS OF SIMULATION **

*(ENTER PROGRAM) ** MEMORY DUMP **

CHAS	SECT				
00	000	44010002	00010002	00000001	77777777

*(END OF MEMORY DUMP) ** POSITIONS OF MEMORY DUMP LISTED CONTAIN NO
INFORMATION PRODUCED BY THE .ASSEMB PROGRAM RUN

SIGNAL TO - LOGS WILL BE TRACED

NO. OF EXECUTIONS SPECIFIED : 5
ENTER INPUT PROGRAM
PREPARE TO OPERATE MODE
SYN BIT COUNTER 1 MODE
SYN BIT COUNTER 2 MODE

G(4) 'G(2)F(1)' IDLE SUB-MODE OF MANUAL HALT
PREPARE TO COMPUTE SUB-MODE OF MANUAL HALT

COMPUTE MODE

TRANSFER INSTRUCTION - (TRA)

CLEAR & ADD INSTRUCTION - (CLA)

A(24-1) = 000 000 000 000 000 000 000 001

ADD INSTRUCTION - (ADD)

A(24-1) = 000 000 000 000 000 000 000 010

ADD INSTRUCTION - (ADD)

A(24-1) = 000 000 000 000 000 000 000 011

NO. OF EXECUTIONS HAVE EXCEEDED NO. SPECIFIED - PROGRAM TERMINATED
TO END REGISTER PROGRAM TYPE BUS ; TO STOP TYPE HALT - BUS

(ENTER PROGRAM) ** PRINTOUT OF INPUT PROGRAM **

REINITIALIZE
02

** RESULTS OF SIMULATION **

MEMORY HAS BEEN INITIALIZED

(ENTER PROGRAM) ** PRINTOUT OF INPUT PROGRAM **

3 ADDITION BUNDLE PROGRAM IS ASCII REPRESENTATION

PROGRAM IN(00) FROM(00) BY(00) NAME

Z 44010002 = 44010002 = 1 = H

MEMORY SIGNAL REGISTER(00) EXECUTE(0000) E(0000) R(00)

PR(0000)

** RESULTS OF SIMULATION **

INPUT SOURCE - NAME TAPE

** MEMORY DUMP **

NAME ADDR

00 000

44010002

44010002

00000001

77777777

** END OF MEMORY DUMP ** (PORTIONS OF MEMORY NOT LISTED CONTAIN NO
INFORMATION PROVIDED BY THE PRESENT PROGRAM RUN)

SIGNAL ON - ADDRS WILL BE TRACED

NO. OF EXECUTIONS SPECIFIED = 5
 MASTER RESET, SEQUENCE
 PREPARE TO OPERATE MODE
 SYNC BIT COUNTER 1 MODE
 SYNC BIT COUNTER 2 MODE

G(4) G(2) G(1) IDLE SUB-MODE OF MANUAL HALT
 PREPARE TO COMPUTE SUB-MODE OF MANUAL HALT

COMPUTE MODE

TRANSFER INSTRUCTIONS - (TFA)

CLEAR & ADD INSTRUCTION - (CLA)

A(24-1) = 000 000 000 000 000 000 000 001

ADD INSTRUCTION - (ADD)

A(24-1) = 000 000 000 000 000 000 000 010

ADD INSTRUCTION - (ADD)

A(24-1) = 000 000 000 000 000 000 000 011

NO. OF EXECUTIONS HAVE BEEN REACHED TO SPECIFIED - PROGRAM TERMINATED
 TO RUN ALTERNATE PROGRAM TYPE 100 : TO STOP TYPE HALT - HALT

END OF PROGRAM
 20.09.19.510P

EXECUTION TIME: 1.125 SEC

Example Program Number 3. This example is an arithmetic program that uses a COA (character output) subroutine to output the answer as eight octal digits. The COA subroutine was developed by the Systems Laboratory Group at Tulane University. (Ref 1:27,28)

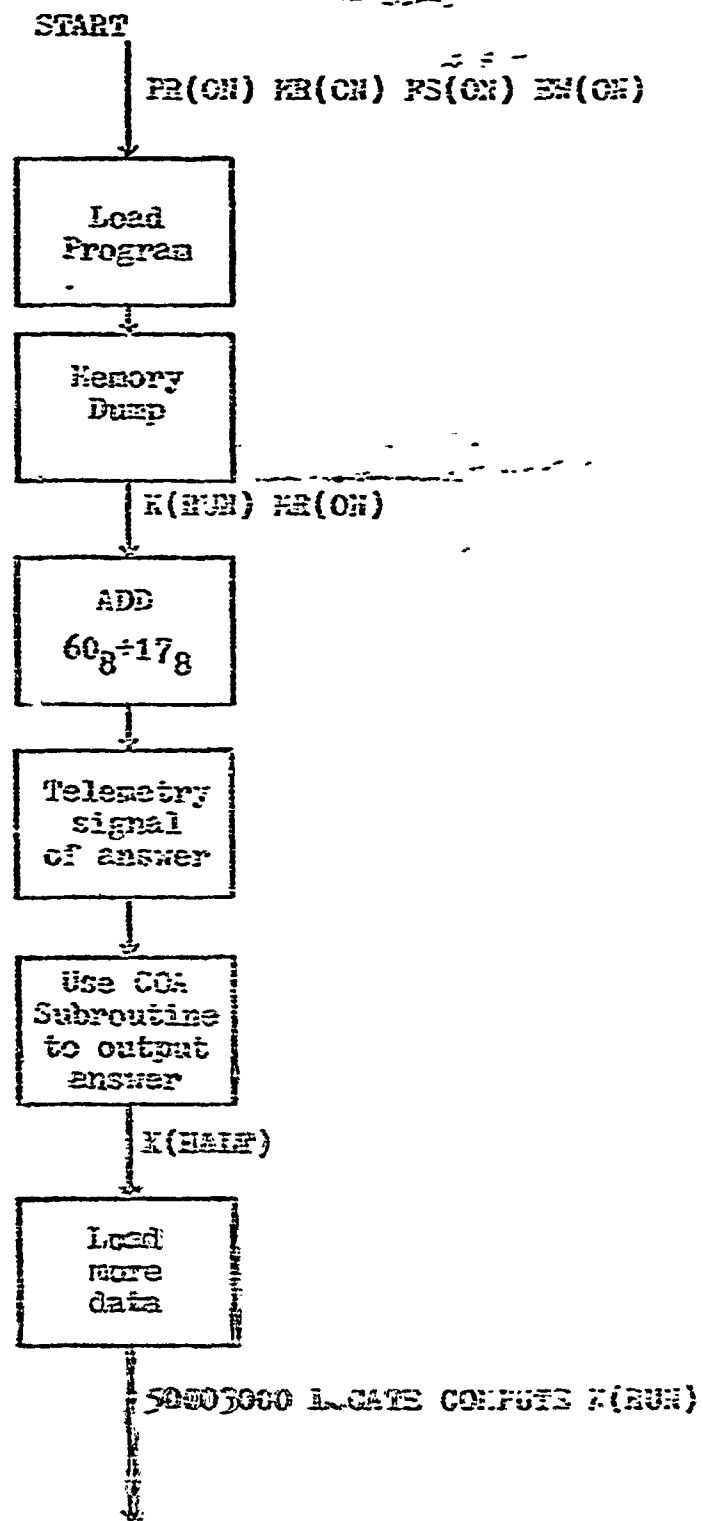
The arithmetic program consists of seven arithmetic routines with starting locations at the program address listed below. Each routine needs two data numbers starting at the data addresses given.

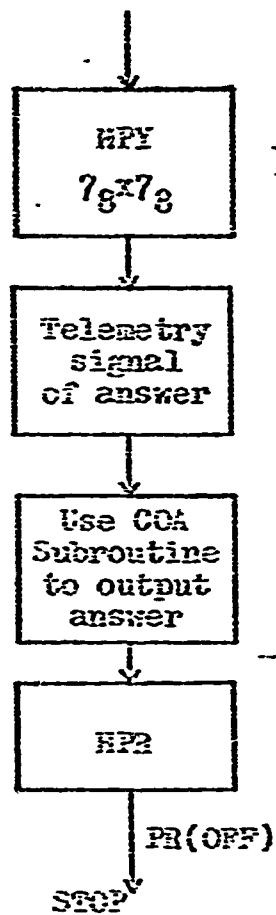
Program Address		Arithmetic Routine	Data Address	
Chan	Sect		Chan	Sect
00	000	ADD routine	02	001
04	000	SUB routine	06	001
10	000	SAD routine	12	001
14	000	SSU routine	16	001
20	000	HPY routine	22	001
24	000	SIF routine	26	001
30	000	Whole No. HPY routine	32	001

Each routine of the arithmetic program can link up with the COA subroutine to output the results. The COA subroutine is loaded in channels 44 and 46. The last instruction of the COA subroutine is an HPR instruction which puts the computer in the program halt mode.

In this example a flag store teleme. signal is also used to output the answer. No mode tracing or register display is used.

The flowchart for example program no. 3 is as follows:





IF OUTPUT IS TO BE DISPLAYED TO PRINTER, TYPE "P" AND "YOUR NAME"; OTHERWISE TYPE "N" - X

```

*****
**                                     **
**                               0173 COMPUTER                               **
**                               SIMULATION PROGRAM                           **
**                                     **
**    DATE= 01/24/72                TIME= 21.00.47                        **
**                                     **
*****

```

(ENTER PROGRAM) ** PRINTOUT OF INPUT PROGRAM **

5 ARITHMETIC PROGRAM WITH CCA SUBROUTINE

```

PR(00) BR(00) SW(00) PS(00) DEAR
2 44010201 = 64020202 = 54034603 = 52404400 =
^ 400 9 44010601 = 74021602 = 54034603 = 52404400 =
^1000 9 44011201 = 81021602 = 54034603 = 52404400 =
^1400 9 44011601 = 71021602 = 54034603 = 52404400 =
^2000 9 44012201 = 61021602 = 54034603 = 52404400 =
^2400 9 44012601 = 51021602 = 54034603 = 52404400 =
^3000 9 44013201 = 41022214 = 47633202 = 60042213 =
24056305 = 54034603 = 52404400 =

```

5 CCA SUBROUTINE

```

^ 4400 9 44014601 = 47024602 = 40034600 = 00072203 = ^ 4407 9 54104610
= 44114601 = 03103201 = ^ 4418 9 47344614 = 40154630 = 03174631 = ^ 4417
9 54204608 = 54014618 = 44224602 = 10234630 = 74044624 = 54254624 =
44264626 = 74274627 = 11144632 = 44314631 = 54324624 = 46332200 = ^ 4601
9 01234567 = ^ 1 = ^ 4614 9 37777777 = ^ 4622 9 ^ 6 = ^ 4624 9 ^ 1 =
^4626 9 ^ 15 = ^ 1 = ^ 4631 9 ^ 6 =

```

```

5 MEMORY CL 001 LDC CL 40 EC CL 17 EN EXEC(1000) BR(00) 2(RUN, 1(HALT)
FILL CL 3201 LDC CL 7 EC CL 7 EN 50030000 LDC CL 01 E(RUN)
PR(OFF)

```

** RESULTS OF SIMULATION **

INPUT SOURCE - MAIN TAPE

** MEMORY DUMP **

CHAS	SECT				
00	000	44010201	64020202	54034603	52404400
04	000	44010601	74020602	54034603	52404400
10	000	44011201	81021202	54034603	52404400
14	000	44011601	71021602	54034603	52404400
20	000	44012201	61022202	54034603	52404400
24	000	44012601	51022602	54034603	52404400
30	000	44013201	41022214	47633202	60042213
30	000	24056305	54034603	52404400	77777777

44	000	44014621	47224502	40034200	00972203
44	004	77777777	77777777	77777777	54194610
44	010	44114631	00133201	77777777	47344614
44	014	40154200	00174021	77777777	64204606
44	020	54214635	44224622	12254430	74244624
44	024	54254624	44264626	74274627	10264430
44	030	44314631	54324624	40332200	77777777
46	000	77777777	01234567	00000001	77777777
46	014	37777777	77777777	77777777	77777777
46	020	77777777	77777777	00000006	77777777
46	024	00000001	77777777	00000015	00000001
46	030	77777777	00000026	77777777	77777777

** END OF MEMORY DUMP ** PORTIONS OF MEMORY NOT LISTED CONTAIN NO INFORMATION PRODUCED BY THE PRESENT PROGRAM RUN

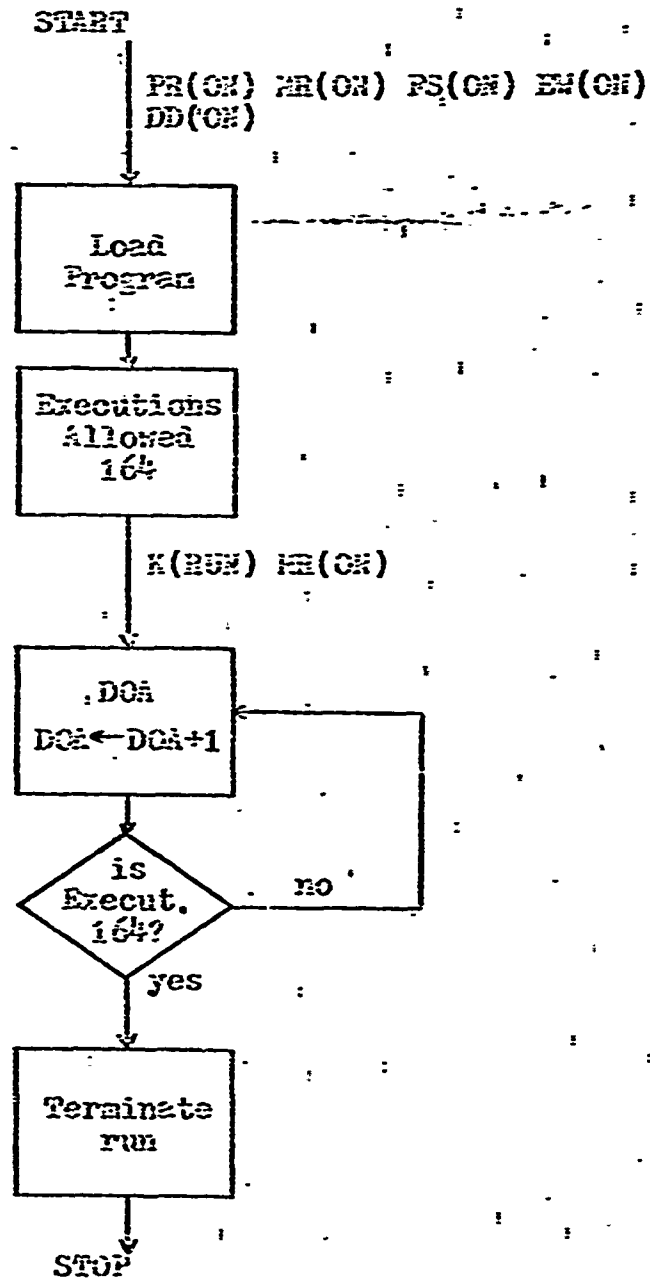
NO. OF EXECUTIONS SPECIFIED = 1000
 FLAGGED INSTRUCTION TELEMETRY SIGNAL - 000000 000000 000000 111111
 BINARY CHARACTER OUTPUT - 0000 HEXIDECIMAL CHARACTER OUTPUT - 0
 BINARY CHARACTER OUTPUT - 0000 HEXIDECIMAL CHARACTER OUTPUT - 0
 BINARY CHARACTER OUTPUT - 0000 HEXIDECIMAL CHARACTER OUTPUT - 0
 BINARY CHARACTER OUTPUT - 0000 HEXIDECIMAL CHARACTER OUTPUT - 0
 BINARY CHARACTER OUTPUT - 0000 HEXIDECIMAL CHARACTER OUTPUT - 0
 BINARY CHARACTER OUTPUT - 0000 HEXIDECIMAL CHARACTER OUTPUT - 0
 BINARY CHARACTER OUTPUT - 0111 HEXIDECIMAL CHARACTER OUTPUT - 7
 BINARY CHARACTER OUTPUT - 0111 HEXIDECIMAL CHARACTER OUTPUT - 7
 FLAGGED INSTRUCTION TELEMETRY SIGNAL - 000000 000000 000000 110001
 BINARY CHARACTER OUTPUT - 0000 HEXIDECIMAL CHARACTER OUTPUT - 0
 BINARY CHARACTER OUTPUT - 0000 HEXIDECIMAL CHARACTER OUTPUT - 0
 BINARY CHARACTER OUTPUT - 0000 HEXIDECIMAL CHARACTER OUTPUT - 0
 BINARY CHARACTER OUTPUT - 0000 HEXIDECIMAL CHARACTER OUTPUT - 0
 BINARY CHARACTER OUTPUT - 0000 HEXIDECIMAL CHARACTER OUTPUT - 0
 BINARY CHARACTER OUTPUT - 0000 HEXIDECIMAL CHARACTER OUTPUT - 0
 BINARY CHARACTER OUTPUT - 0110 HEXIDECIMAL CHARACTER OUTPUT - 6
 BINARY CHARACTER OUTPUT - 0001 HEXIDECIMAL CHARACTER OUTPUT - 1
 TO RUN ADDRESS PROGRAM TYPE "RUN"; TO STOP TYPE "HALT" - HALT

** END OF PROGRAM
 21.13.20 STOP

EXECUTION TIME: 1.721 SEC

Example Program Number 4. This example is a program which shows the discrete output capability of the simulation program. To interpret the discrete output listing, the reader should refer to Fig. 12 in Appendix C.

The flowchart for example program no. 4 is as follows:



GE/EE/72-7

IF OUTPUT IS TO BE DISPLAYED TO PRINTER, TYPE "P" AND "YOUR NAME"; OTHERWISE TYPE "A" - "A"

```
*****
**
**          DIB COMPUTER          **
**      SIMULATION PROGRAM      **
**
**      DATE= 01/24/72          TIME= 19.16.37
**
*****
```

** PRINTOUT OF INPUT PROGRAM **

(ENTER PROGRAM)

3 DISCRETE OUTPUT PROGRAM

PR(00) NR(00) EX(00) FILL
: 40012630 EN 44023000 EN 64033205 EN 84040302 EN 50000000 EN
CL 205 LOC CL 11 EN EXEC(0164) EX(00) NR(00)
PR(077)

** RESULTS OF SIMULATION **

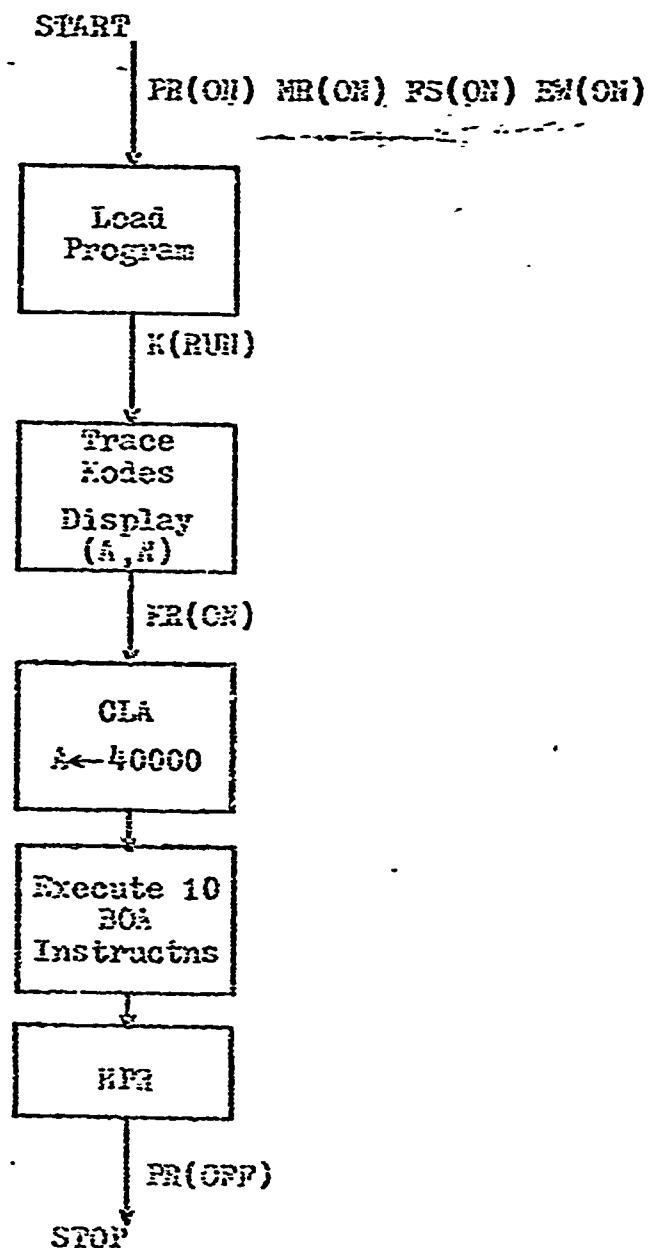
NO. OF EXECUTIONS SPECIFIED = 164
DISCRETE OUTPUT LINE 0 10 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 20 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 30 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 40 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 50 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 60 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 70 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 80 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 90 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 100 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 110 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 120 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 130 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 140 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 150 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 160 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 170 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 180 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 190 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 200 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 210 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 220 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 230 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 240 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 250 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 260 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 270 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 280 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 290 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 300 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 310 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 320 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 330 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 340 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 350 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 360 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 370 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 380 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 390 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 400 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 410 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 420 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 430 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 440 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 450 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 460 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 470 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 480 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 490 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 500 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 510 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 520 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 530 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 540 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 550 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 560 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 570 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 580 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 590 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 600 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 610 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 620 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 630 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 640 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 650 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 660 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 670 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 680 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 690 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 700 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 710 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 720 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 730 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 740 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 750 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 760 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 770 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 780 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 790 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 800 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 810 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 820 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 830 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 840 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 850 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 860 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 870 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 880 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 890 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 900 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 910 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 920 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 930 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 940 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 950 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 960 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 970 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 980 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 990 HAS A 1 OUTPUT SIGNAL
DISCRETE OUTPUT LINE 0 1000 HAS A 1 OUTPUT SIGNAL
NO. OF EXECUTIONS NOT SPECIFIED - PROGRAM TERMINATED
TO END AFTER PRINTING TYPE "A"; TO STOP TYPE "HALT" - HALT

END OF PROGRAM
19.22.36.817

EXECUTION TIME= .552 SEC

Example Program Number 5. The program for this example uses the binary output instructions (BOA, BOB, BOC) and shows the binary output capability of the simulation program. To interpret the binary output listing, the reader should refer to Fig. 13 in Appendix C.

The flowchart for example program no. 5 is as follows:



IF OUTPUT IS TO BE DISPOSED TO PRINTER, TYPE "P" AND "YOUR NAME"; OTHERWISE TYPE "N" - N

```

*****
**                                     **
**               DITS COMPUTER          **
**            SIMULATION PROGRAM        **
**                                     **
**   DATE= 01/24/72                TIME= 20.10.14   **
**                                     **
*****

```

*(ENTER PROGRAM) ** PRINTOUT OF INPUT PROGRAM **

S BINARY OUTPUT PROGRAM

```

PR(00) ER(00) FS(00) EX(00) FILL
40010201 EN 40021000 EN 40031000 EN 40041000 EN 40051000 EN 40061000 EN
40071000 EN 40081000 EN 40091000 EN 40101000 EN 40111000 EN 40121000 EN 40131000 EN 40002200 EN
CL 201 LCC 00400000 EN Z(000) SIGNAL REG(A,B) ER(00)
PR(0FF)

```

** RESULTS OF SIMULATION **

SIGNAL 00 - MODES WILL BE TRACED

FASTER RESET SEQUENCE
 PREPARE TO OPERATE MODE
 SYNC BIT COUNTER 1 MODE
 SYNC BIT COUNTER 2 MODE

S(4) 042)0(1) IDLE SUB-MODE OF MANUAL HALT
 PREPARE TO COMPUTE SUB-MODE OF MANUAL HALT

COMPUTE MODE

TRANSFER INSTRUCTION - (TRA)

```

Z(24-1) = 000 000 100 000 000 000 000 000
CLEAR & ADD INSTRUCTION - (CLA)
A(24-1) = 000 000 100 000 000 000 000 000

```

```

BINARY OUTPUT "A" INSTRUCTION - (BCA)
BINARY OUTPUT ON LINE 610 OF +1
A(24-1) = 000 000 010 000 000 000 000 000

```

```

BINARY OUTPUT "A" INSTRUCTION - (BCA)
BINARY OUTPUT ON LINE 610 OF +1
A(24-1) = 000 000 000 000 000 000 000 000

```

```

BINARY OUTPUT "A" INSTRUCTION - (BCA)
BINARY OUTPUT ON LINE 611 OF -1
A(24-1) = 111 111 110 000 000 000 000 000

```

BINARY OUTPUT "A" INSTRUCTION - (G2A)
 BINARY OUTPUT ON LINE G10 OF +1
 A(24-1) = 000 000 000 000 000 000 000

BINARY OUTPUT "A" INSTRUCTION - (G2A)
 BINARY OUTPUT ON LINE G11 OF -1
 A(24-1) = 111 111 110 000 000 000 000

BINARY OUTPUT "A" INSTRUCTION - (G2A)
 BINARY OUTPUT ON LINE G10 OF +1
 A(24-1) = 000 000 000 000 000 000 000

BINARY OUTPUT "A" INSTRUCTION - (G2A)
 BINARY OUTPUT ON LINE G11 OF -1
 A(24-1) = 111 111 110 000 000 000 000

BINARY OUTPUT "A" INSTRUCTION - (G2A)
 BINARY OUTPUT ON LINE G10 OF +1
 A(24-1) = 000 000 000 000 000 000 000

BINARY OUTPUT "A" INSTRUCTION - (G2A)
 BINARY OUTPUT ON LINE G11 OF -1
 A(24-1) = 111 111 110 000 000 000 000

BINARY OUTPUT "A" INSTRUCTION - (G2A)
 BINARY OUTPUT ON LINE G10 OF +1
 A(24-1) = 000 000 000 000 000 000 000

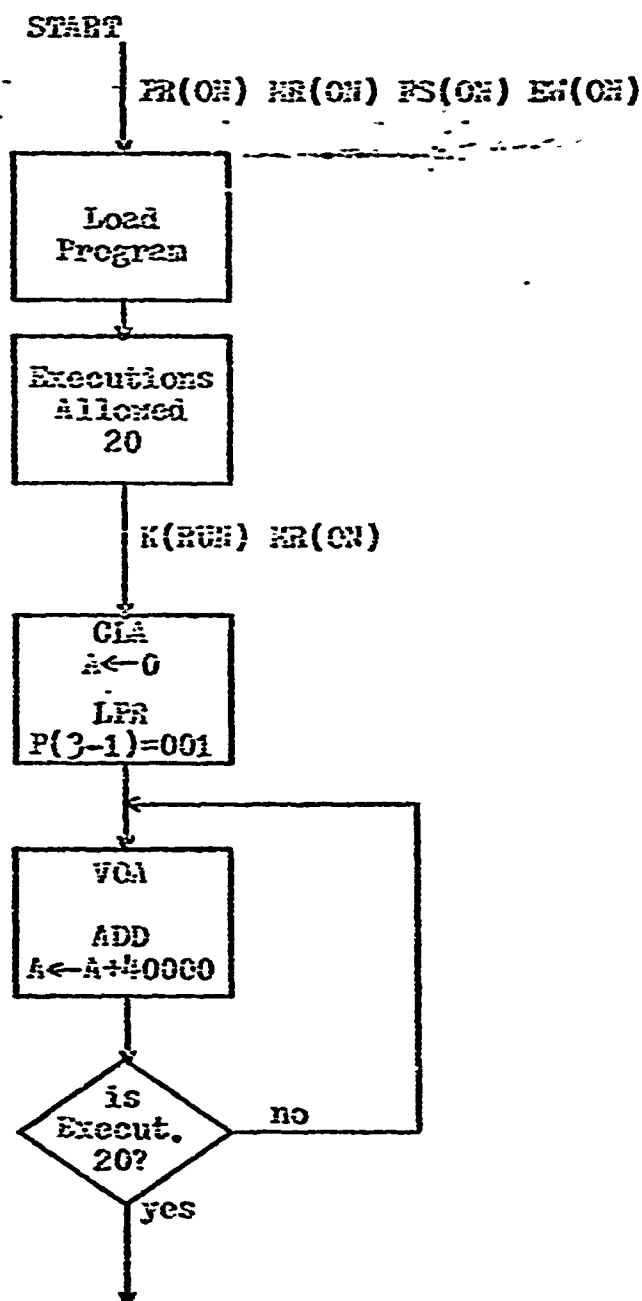
HALT AND PROCEED INSTRUCTION - (GPE)
 PROGRAM HALT EDGE

POWER HAS BEEN TURNED OFF
 TO RUN ANOTHER PROGRAM TYPE "RUN"; TO STOP TYPE "HALT" - HALT

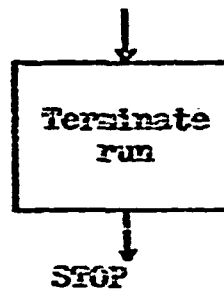
** END OF PROGRAM EXECUTION TIME: .659 SEC
 20.16.11 STOP

Example Program Number 6. This example is a program which uses the voltage output instructions (VQA, VQB, VQC) and shows the voltage output capability of the simulation program. To interpret the voltage output listing, the reader should refer to Fig. 14 in Appendix C.

The flowchart for this example is as follows:



GE/EE/72-7



GE/EE/72-7

IF OUTPUT IS TO BE DISPOSED TO PRINTER, TYPE "P" AND "YOUR NAME"; OTHERWISE TYPE "N" - N

```
*****
**
**                               **
**          DITS COMPUTER        **
**        SIMULATION PROGRAM     **
**
**    DATE= 02/07/72             **
**                               **
**    TIME= 20.40.04             **
**                               **
*****
```

** PRINTOUT OF INPUT PROGRAM **

(ENTER PROGRAM)

S VOLTAGE OUTPUT PROGRAM

PR(08) HR(02) FS(08) EX(08) FILL
40017200 EN 44020201 EN 40033000 EN 64020202 EN 50000002 EN
CL 201 LCC CL EN CL 200000 EN EXEC(0020) X(RUN) HR(09)
PR(0FF)

** RESULTS OF SIMULATION **

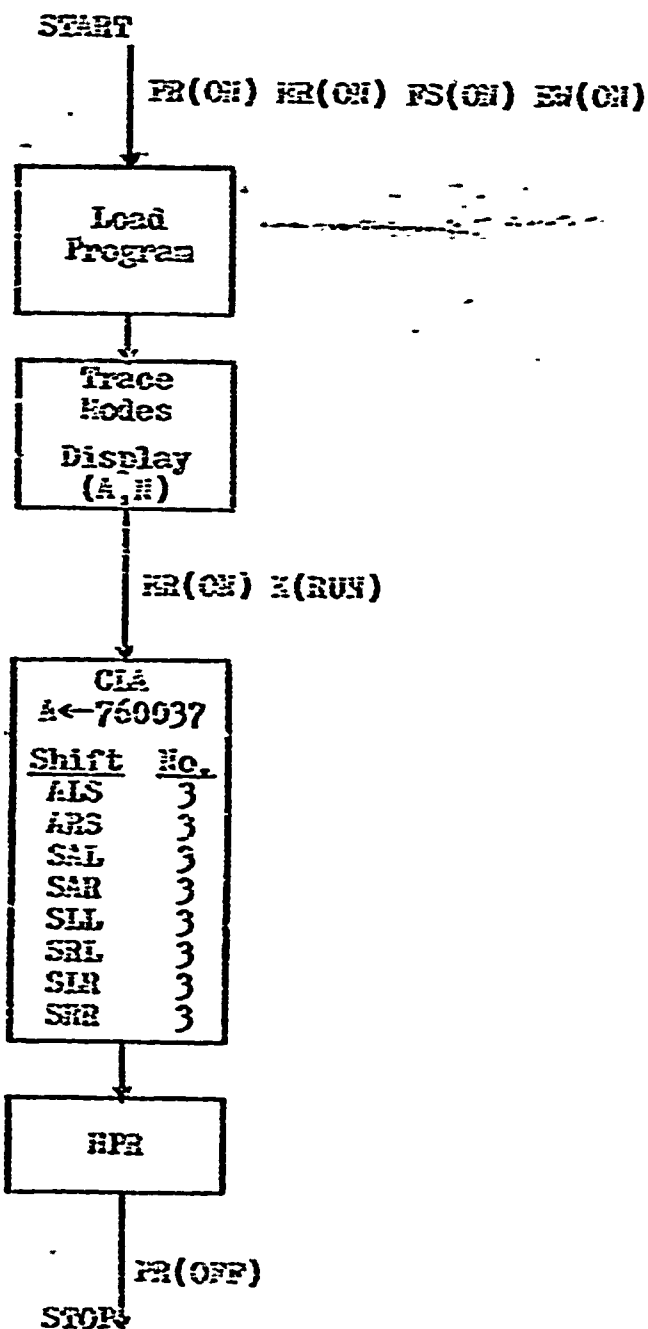
NO. OF EXECUTIONS SPECIFIED = 20
PHASE REGISTER - P(3-1) = 001
VI(8-1) = 00000000 WITH A VOLTAGE OUTPUT OF 0.00 VOLTS
VOLTAGE OUTPUT IS ON LINE V011
VI(8-1) = 00000001 WITH A VOLTAGE OUTPUT OF .16 VOLTS
VOLTAGE OUTPUT IS ON LINE V011
VI(8-1) = 00000010 WITH A VOLTAGE OUTPUT OF .31 VOLTS
VOLTAGE OUTPUT IS ON LINE V011
VI(8-1) = 00000011 WITH A VOLTAGE OUTPUT OF .47 VOLTS
VOLTAGE OUTPUT IS ON LINE V011
VI(8-1) = 00000100 WITH A VOLTAGE OUTPUT OF .63 VOLTS
VOLTAGE OUTPUT IS ON LINE V011
VI(8-1) = 00000101 WITH A VOLTAGE OUTPUT OF .78 VOLTS
VOLTAGE OUTPUT IS ON LINE V011
VI(8-1) = 00000110 WITH A VOLTAGE OUTPUT OF .94 VOLTS
VOLTAGE OUTPUT IS ON LINE V011
VI(8-1) = 00000111 WITH A VOLTAGE OUTPUT OF 1.09 VOLTS
VOLTAGE OUTPUT IS ON LINE V011
NO. OF EXECUTIONS HAVE EXCEEDED NO. SPECIFIED - PROGRAM TERMINATED
TO RUN ANOTHER PROGRAM TYPE "RUN"; TO STOP TYPE "HALT" - HALT

** END OF PROGRAM
20.45.04.STOP

EXECUTION TIME= .771 SEC

Example Program Number 7. This program is an example of a program which uses all the shift instructions of the D17B computer instruction set. Mode tracing and register display are used in the compute mode only.

The flowchart for example program no. 7 is as follows:



IF OUTPUT IS TO BE DISPOSED TO PRINTER, TYPE "P" AND "YOUR NAME"; OTHERWISE TYPE "N" - N

```

**-----**
**                               **
**             DIB COMPUTER      **
**          SIMULATION PROGRAM    **
**                               **
**   DATE= 01/24/72             TIME= 19.36.25 **
**                               **
**-----**

```

(ENTER PROGRAM) ** PRINTOUT OF INPUT PROGRAM **

S SHIFT PROGRAM

```

PR(02) ER(02) PS(02) EV(02) FILL
44016201 EN 00022205 EN 03033204 EN 00042005 EN 00053003 EN
00062403 EN 00072605 EN 03103403 EN 00115505 EN 40022203 EN
CL 201 LDC 00760037 EN SIGNAL REGISTER(R,S) E(RUS) ER(02)
PR(OFF)

```

** RESULTS OF SIMULATION **

SIGNAL 02 - MODE WILL BE TRACED

```

FASTER RESET SEQUENCE
PREPARE TO OPERATE MODE
SYNC BIT COUNTER 1 MODE
SYNC BIT COUNTER 2 MODE

```

```

S(4) 0(2)0(1) IDLE SCS-MODE OF RALCAL HALT
PREPARE TO COMPUTE SCS-MODE OF RALCAL HALT

```

COMPUTE MODE

TRANSFER INSTRUCTION - (TEN)

```

E(24-1) = 000 000 111 110 000 000 011 111
CLEAR & ADD INSTRUCTION - (CLA)
A(24-1) = 000 000 111 110 000 000 011 111

```

```

ACCUMULATOR LEFT SHIFT INSTRUCTION - (ALS)
A(24-1) = 011 111 000 000 001 111 100 000

```

```

ACCUMULATOR RIGHT SHIFT INSTRUCTION - (ARS)
A(24-1) = 000 001 111 100 000 000 111 110

```

```

SPLIT ACCUMULATOR LEFT SHIFT INSTRUCTION - (SAL)
A(24-1) = 001 111 100 007 700 111 110 000

```

```

SPLIT ACCUMULATOR RIGHT SHIFT INSTRUCTION - (SAR)
A(24-1) = 000 001 111 107 700 000 111 110

```

GE/EE/72-7

SPLIT LEFT WORD LEFT SHIFT INSTRUCTION - (SL1)
A(24-1) = 001 111 100 007 700 000 111 110

SPLIT RIGHT WORD LEFT SHIFT INSTRUCTION - (SL2)
A(24-1) = 001 111 100 007 700 111 110 000

SPLIT LEFT WORD RIGHT SHIFT INSTRUCTION - (SL2)
A(24-1) = 000 001 111 107 700 111 110 000

SPLIT RIGHT WORD RIGHT SHIFT INSTRUCTION - (SL2)
A(24-1) = 000 001 111 107 700 000 111 110

HALT AND PROCEED INSTRUCTION - (SP2)
PROGRAM HALT MODE

POWER HAS BEEN TURNED OFF
TO RUN AGAIN PROGRAM TYPE "RST"; TO STOP TYPE "HOLD" - HALT

** END OF PROGRAM
19.41.59.572

EXECUTION TIME = .317 SEC

VI. Conclusion

A software simulation program of the ~~Minuteman~~ D17B Computer was written to simulate the functions of the D17B computer. The objectives of this simulation were to have the simulation program simulate the actual computer as closely as possible. This objective was met because the majority of the D17B functions have been included in the simulation program. The loading and interaction functions of the noncompute mode have been used. In the compute mode, the searching, reading and writing memory, and instruction execution are all part of the simulation program. Wherever possible, the same algorithm implemented on the D17B was used in the simulation program. This approach resulted in some inefficiencies in the simulation program, but a by-product of using the same algorithm is that the simulation program can be used as a teaching aid for learning the operation of the D17B computer. Also error detection was built into the simulation program and has been very helpful in creating program tapes for the D17B computer.

Recommendations for Future Study. There are some D17B computer functions which have not been incorporated in the simulation program. Two of these functions are associated with real time control processing and include the capabilities for incremental inputs and fine count-down operations.

The D17B computer is capable of detecting and

incrementally adding bits of information to the words of the V-loop and R-loop without program control. Also when the EFC (enter fine countdown) instruction is executed, the computer goes into the fine countdown mode. In the fine countdown mode, the V-loop and U-loop are linked together forming a digital integrator which operates without program control.

The incremental input and fine countdown mode functions listed above would be important if the simulation were to be used for real time control. Since this was not the original purpose of the simulation program, these functions are not included. However, the fine countdown instructions (HFC-halt fine countdown and EFC) and the instructions which store and unload information from the V-loop and R-loop are a part of the simulation program. Also, a subroutine for storing incremental data supplied by the user into the V-loop and R-loop is a part of the simulation program. The incremental input and fine countdown mode capabilities described above are improvements that could be added by a thesis student who is researching the area of real time control applications for the D17B computer.

Another recommendation for future study would be the creation of an assembler for the D17B computer. The assembler could be written for operation on the CDC 6600 computer. The assembler would accept as input a program written in D17B masonic coding and output on punched tape a machine

GE/EE/72-7

language version of the program. This machine language program on the punched tape could then be supplied as data to both the D17B computer and the simulation program.

Bibliography

1. Beck, C. R. D172 Computer Programming Manual. Report HSCG-4-71. New Orleans, Louisiana: Tulane University Systems Laboratory, Department of Electrical Engineering, September 1971.
2. Chu, Y. Introduction to Computer Organization. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1970.
3. Control Data Corporation. Intercom 2 Reference Manual. Publication No. 60306000. Sunnyvale, California: CDC, 1970.
4. Hansen D. D. and Watkins R. R. A Rigorous Logical Study With Lab of the D17 Digital Computer. AOC-311702-33. Computers and Data Systems Logistics, Customer Training, 30 April 1962.
5. ----- Minuteman D17 Computer Training Data. Autonetics, a division of North American Rockwell, 8 June 1970.
6. ----- Preliminary Maintenance Manual of the Minuteman D17 Computer and Associated Test Equipment. Project Office Memorandum No. 71. Autonetics, a division of North American Aviation, Inc., January 1960.
7. Shroyer, L. G. D17 Computer Manual. Field Engineering (N2 Logistics), 1 July 1960.
8. T. O. 1102-10-5-3-5. General Purpose Digital Computer (Model D173). Technical Manual, Overhaul, Autonetics, 1 December 1963, changed 12 October 1964.

VITA

Bruce Chatterton was born on 31 March 1940 in Franklin, Idaho. He graduated from high school in Preston, Idaho in 1958 and attended Utah State University for five quarters. He enlisted in the USAF in December 1962 and received 36 weeks of training in electronics and communication equipment repair at Sheppard AFB, Texas. While stationed at McClellan AFB, California, he attended American River Junior College to become eligible for education under the Airman Education and Commission Program (AECOP). He was accepted for AECOP training in June 1965 and attended Oklahoma State University where he received the degree of Bachelor of Science in Electrical Engineering in July 1967. He then attended Officer Training School at Lackland AFB, Texas and received a commission in the USAF in November 1967. He served as a project officer for the Air Force Satellite Control Facility in Los Angeles, California. He attended the Air Force Institute of Technology where he received the degree of Master of Science in Electrical Engineering in March 1972.

Permanent Address: P.O. Box 177
Franklin, Idaho 83237

This thesis was typed by Bruce Chatterton

GE/EE/72-7

Appendix A

Printout of Simulation Program

Line	Code	Text	Address
1	C	***** INPUT, OUTPUT, TIME=TIME, TIME=TIME, TIME=TIME	1
2	C	*****	2
3	C	*****	3
4	C	*****	4
5	C	*****	5
6	C	*****	6
7	C	*****	7
8	C	*****	8
9	C	*****	9
10	C	*****	10
11	C	*****	11
12	C	*****	12
13	C	*****	13
14	C	*****	14
15	C	*****	15
16	C	*****	16
17	C	*****	17
18	C	*****	18
19	C	*****	19
20	C	*****	20
21	C	*****	21
22	C	*****	22
23	C	*****	23
24	C	*****	24
25	C	*****	25
26	C	*****	26
27	C	*****	27
28	C	*****	28
29	C	*****	29
30	C	*****	30
31	C	*****	31
32	C	*****	32
33	C	*****	33
34	C	*****	34
35	C	*****	35
36	C	*****	36
37	C	*****	37
38	C	*****	38
39	C	*****	39
40	C	*****	40
41	C	*****	41
42	C	*****	42
43	C	*****	43
44	C	*****	44
45	C	*****	45
46	C	*****	46
47	C	*****	47
48	C	*****	48
49	C	*****	49
50	C	*****	50
51	C	*****	51
52	C	*****	52
53	C	*****	53
54	C	*****	54
55	C	*****	55
56	C	*****	56
57	C	*****	57
58	C	*****	58
59	C	*****	59
60	C	*****	60
61	C	*****	61
62	C	*****	62
63	C	*****	63
64	C	*****	64
65	C	*****	65
66	C	*****	66
67	C	*****	67
68	C	*****	68
69	C	*****	69
70	C	*****	70
71	C	*****	71
72	C	*****	72
73	C	*****	73
74	C	*****	74
75	C	*****	75
76	C	*****	76
77	C	*****	77
78	C	*****	78
79	C	*****	79
80	C	*****	80
81	C	*****	81
82	C	*****	82
83	C	*****	83
84	C	*****	84
85	C	*****	85
86	C	*****	86
87	C	*****	87
88	C	*****	88
89	C	*****	89
90	C	*****	90
91	C	*****	91
92	C	*****	92
93	C	*****	93
94	C	*****	94
95	C	*****	95
96	C	*****	96
97	C	*****	97
98	C	*****	98
99	C	*****	99
100	C	*****	100

A-2

```

50 TO 135
C
COMMENT **TRANSLATION OF LOAD CODES
135 IF (P1=177.00.1) GO TO 136
DO 145 I1=1,5
145 IT(I1)=1
IT(I1)=1
IF (MCP2(MCCL.50.1M(19).C2.MCOP2(MCCL.50.1M(11)) IT(5)=1
IF (MCP2(MCCL.50.1M(15)) IT(5)=1
IF (MCP2(MCCL.50.1M(21).M2.MCOP2(MCCL.50.1M(13)) IT(5)=1
IF (MCP2(MCCL.50.1M(13).C2.MCOP2(MCCL.50.1M(14)) IT(2)=2
IF (MCP2(MCCL.50.1M(15)) IT(2)=IT(2)+2
IF (MCP2(MCCL.50.1M(19).C2.MCOP2(MCCL.50.1M(21)) IT(2)=1
IF (MCP2(MCCL.50.1M(21).M2.MCOP2(MCCL.50.1M(13)) IT(2)=1
IF (MCP2(MCCL.50.1M(13).C2.MCOP2(MCCL.50.1M(21)) IT(1)=1
IF (MCP2(MCCL.50.1M(14).C2.MCOP2(MCCL.50.1M(15)) IT(1)=1
145 MCCL=MCCL+1
IF (MCCL.50.72) GO TO 135
IF (MCP2(MCCL.50.1M(19)) GO TO 135
GO TO 145
155 WRITE(6,2145)
WRITE(6,2145)
GO TO 155
C
COMMENT **TRANSLATION OF PRIMARY INPUT DATA
155 I2=MCCL
DO 161 I1=1,6
IF (MCP1(I2).M2.MCOP1(I2).M2.MCOP1(I2)) GO TO 156
IT(I1-I1)=1
IF (MCP1(I2).50.1M(21)) IT(5-I1)=1
I2=I2+1
161 CONTINUE
IF (MCP1(MCCL.50.1M(19)) GO TO 165
MCCL=MCCL+5
GO TO 165
165 DO 171 I1=MCCL,72
IF (MCP1(I1).50.1M(19)) GO TO 175
171 CONTINUE
175 WRITE(6,2155) MCP1(I1),I2=MCCL,I1
WRITE(6,2155) MCP1(I1),I2=MCCL,I1
MCCL=I1
GO TO 165
C
COMMENT **TRANSLATION OF TIMING SIGNAL
185 MCCL=MCCL+1
IF (MCCL.50.72) GO TO 195
IF (MCP2(MCCL.50.1M(19)) GO TO 195
GO TO 195
195 I=0
IF (I.50.00) GO TO 197
IF (I=50.00) GO TO 197
IF (I=50.01) GO TO 195
WRITE(6,2155)
WRITE(6,2155)
GO TO 195
197 WRITE(6,2155)
WRITE(6,2155)
GO TO 195
C
COMMENT **TRANSLATION OF COMPUTE MODE SWITCH
205 IF (MCP1(MCCL.2).50.1M(23)) GO TO 210
IF (MCP1(MCCL.2).50.1M(23)) GO TO 210
IF (MCP1(MCCL.2).50.1M(24)) GO TO 210
WRITE(6,2165)
GO TO 165
210 C=5160
GO TO 215
215 W=MLT
GO TO 215
217 W=0
215 MCCL=MCCL+1
IF (MCCL.50.72) GO TO 220
IF (MCP2(MCCL.50.1M(19)) GO TO 220
GO TO 215
220 IF (I=50.00.00).K=50.00) GO TO 420
IF (I=50.00.00).K=50.00) GO TO 575
IF (I=50.00.00).K=50.00) GO TO 600
IF (I=50.00.00).K=50.00) GO TO 565
GO TO 45
C
COMMENT **COMPUTE RESULTS
225 MCCL=MCCL+1

```

Reproduced from
document available

SI ¹⁶	167
SI ¹⁷	168
SI ¹⁸	169
SI ¹⁹	170
SI ²⁰	171
SI ²¹	172
SI ²²	173
SI ²³	174
SI ²⁴	175
SI ²⁵	176
SI ²⁶	177
SI ²⁷	178
SI ²⁸	179
SI ²⁹	180
SI ³⁰	181
SI ³¹	182
SI ³²	183
SI ³³	184
SI ³⁴	185
SI ³⁵	186
SI ³⁶	187
SI ³⁷	188
SI ³⁸	189
SI ³⁹	190
SI ⁴⁰	191
SI ⁴¹	192
SI ⁴²	193
SI ⁴³	194
SI ⁴⁴	195
SI ⁴⁵	196
SI ⁴⁶	197
SI ⁴⁷	198
SI ⁴⁸	199
SI ⁴⁹	200
SI ⁵⁰	201
SI ⁵¹	202
SI ⁵²	203
SI ⁵³	204
SI ⁵⁴	205
SI ⁵⁵	206
SI ⁵⁶	207
SI ⁵⁷	208
SI ⁵⁸	209
SI ⁵⁹	210
SI ⁶⁰	211
SI ⁶¹	212
SI ⁶²	213
SI ⁶³	214
SI ⁶⁴	215
SI ⁶⁵	216
SI ⁶⁶	217
SI ⁶⁷	218
SI ⁶⁸	219
SI ⁶⁹	220
SI ⁷⁰	221
SI ⁷¹	222
SI ⁷²	223
SI ⁷³	224
SI ⁷⁴	225
SI ⁷⁵	226
SI ⁷⁶	227
SI ⁷⁷	228
SI ⁷⁸	229
SI ⁷⁹	230
SI ⁸⁰	231
SI ⁸¹	232
SI ⁸²	233
SI ⁸³	234
SI ⁸⁴	235
SI ⁸⁵	236
SI ⁸⁶	237
SI ⁸⁷	238
SI ⁸⁸	239
SI ⁸⁹	240
SI ⁹⁰	241
SI ⁹¹	242
SI ⁹²	243
SI ⁹³	244
SI ⁹⁴	245
SI ⁹⁵	246
SI ⁹⁶	247
SI ⁹⁷	248

```

IF(FCFL(CT,72)) GO TO 49
IF((COP)CFL(0001).EQ.MPL(0001) GO TO 45
GO TO 225

C
COMMENT **TRANSLATION OF JUMP/CALL RELOCATION
224 CTR=1
GO TO 225
225 CTR=C
GO TO 225

C
COMMENT **TRANSLATION OF SIGNAL COMMAND
240 IF(SIGNAL(0001) EQ 0C 245
SIGNAL=1
WRITE(6,227(1)
GO TO 225
245 SIGNAL=3
WRITE(5,2175)
GO TO 225

C
COMMENT **TRANSLATION OF MARK CODE DESIGNATION
250 MARK=1
WRITE(6,2182)
GO TO 225
255 MARK=3
GO TO 225

C
COMMENT **TRANSLATION OF REGISTER DISPLAY COMMAND
260 CALL REG1
GO TO 45

C
COMMENT **TRANSLATION OF MEMORY DISPLAY COMMAND
265 CALL MEMOY
GO TO 45

C
COMMENT **TRANSLATION OF INCREMENTAL & DISCRETE TABLES
270 CALL DIS X
GO TO 45
275 CALL DIS Y
GO TO 45
280 CALL INC R
GO TO 45
285 CALL INC V
GO TO 45

C
COMMENT **TRANSLATION OF DISCRETE SPECIAL FLIPFLOP
290 DFL=1
GO TO 225

C
COMMENT **TRANSLATION OF EVENT SPECIFICATION
305 CONTINUE
310 SIGNAL=0001
IF(FCFL(CT,72)) GO TO 45
IF((COP)CFL(0001).EQ.MPL(0001) GO TO 315
GO TO 310
315 DO 315 I=1,4
PG 31. IZ=1,1
IF((COP)CFL(0001).EQ.MPL(0001) GO TO 315
320 CONTINUE
WRITE(5,2192)
WRITE(5,2193)
GO TO 225
325 C=C+(I+1)*12-1
LIST=LIST*(C-C+1)+1.*PC*25(12)+13.*EC*25(13)+C2*25(14)
WRITE(5,2195) LIST
WRITE(4,2195) LIST
GO TO 225

C
COMMENT **TRANSLATION OF PC & MC FLIPFLOP SETTINGS
330 IF((COP)CFL(0001).EQ.MPL(0001) GO TO 325
DFF=1
GO TO 225
335 DFF=1
GO TO 225
340 IF((COP)CFL(0001).EQ.MPL(0001) GO TO 335
DFF=1
GO TO 225
345 DFF=1
GO TO 225

C
COMMENT **TRANSLATION OF FILL SWITCH
350 SIGNAL=0001
IF(FCFL(CT,72)) GO TO 45
IF((COP)CFL(0001).EQ.MPL(0001) GO TO 45

```

512	269
512	259
512	251
512	252
512	253
514	254
512	255
512	256
512	257
512	258
512	259
512	260
512	261
512	262
512	263
512	264
512	265
512	266
512	267
512	268
512	269
512	270
512	271
512	272
512	273
512	274
512	275
512	276
512	277
512	278
512	279
512	280
512	281
512	282
512	283
512	284
512	285
512	286
512	287
512	288
512	289
512	290
512	291
512	292
512	293
512	294
512	295
512	296
512	297
512	298
512	299
512	300
512	301
512	302
512	303
512	304
512	305
512	306
512	307
512	308
512	309
512	310
512	311
512	312
512	313
512	314
512	315
512	316
512	317
512	318
512	319
512	320
512	321
512	322
512	323
512	324
512	325
512	326
512	327
512	328
512	329
512	330
512	331

GO TO 349	SIN	332
345 FS=ON	SIN	333
IF (FS=ON) GO TO 349	SIN	334
IF (FS=ON) GO TO 378	SIN	335
GO TO 45	SIN	336
C	SIN	337
COMMENT **TRANSLATION OF COLD-STOPPAGE WRITE SWITCH	SIN	338
355 IF (W0000000000000000) GO TO 355	SIN	339
IF (W0000000000000000) GO TO 369	SIN	340
WRITE(15,2132)	SIN	341
GO TO 165	SIN	342
375 SW=ON	SIN	343
GO TO 225	SIN	344
363 SW=OFF	SIN	345
GO TO 225	SIN	346
C	SIN	347
COMMENT **TRANSLATION OF DISCRETE SWITCH	SIN	348
375 IF (W0000000000000000) GO TO 375	SIN	349
IF (W0000000000000000) GO TO 375	SIN	350
WRITE(16,2135)	SIN	351
GO TO 165	SIN	352
377 SW=ON	SIN	353
GO TO 225	SIN	354
375 SW=OFF	SIN	355
GO TO 225	SIN	356
C	SIN	357
COMMENT **TRANSLATION OF MECHANICAL INPUT SWITCH	SIN	358
375 SW=ON	SIN	359
IF (W0000000000000000) GO TO 375	SIN	360
IF (W0000000000000000) GO TO 365	SIN	361
GO TO 332	SIN	362
375 SW=ON	SIN	363
IF (W0000000000000000) GO TO 349	SIN	364
GO TO 45	SIN	365
C	SIN	366
COMMENT **STOP-ON/OFF SEQUENCE	SIN	367
375 IF (W0000000000000000) GO TO 437	SIN	368
IF (W0000000000000000) GO TO 365	SIN	369
WRITE(15,2132)	SIN	370
GO TO 165	SIN	371
375 SW=OFF	SIN	372
IF (W0000000000000000) WRITE(15,2135)	SIN	373
WRITE(15,2135)	SIN	374
GO TO 15	SIN	375
377 SW=ON	SIN	376
IF (W0000000000000000) WRITE(15,2132)	SIN	377
GO TO 375	SIN	378
C	SIN	379
375 WRITE(15,2135)	SIN	380
WRITE(15,2135)	SIN	381
GO TO 15	SIN	382
377 WRITE(15,2135)	SIN	383
WRITE(15,2135)	SIN	384
GO TO 15	SIN	385
375 WRITE(15,2135)	SIN	386
WRITE(15,2135)	SIN	387
GO TO 15	SIN	388
377 WRITE(15,2135)	SIN	389
WRITE(15,2135)	SIN	390
GO TO 15	SIN	391
375 WRITE(15,2135)	SIN	392
WRITE(15,2135)	SIN	393
GO TO 1	SIN	394
375 WRITE(15,2135)	SIN	395
WRITE(15,2135) VOLTAGE	SIN	396
WRITE(15,2135) VOLTAGE	SIN	397
STOP	SIN	398
C	SIN	399
*****	SIN	400
*****	SIN	401
*****	SIN	402
*****	SIN	403
*****	SIN	404
C	SIN	405
COMMENT **STOPPED RESET SEQUENCE	SIN	406
375 IF (W0000000000000000) GO TO 417	SIN	407
375 SW=ON	SIN	408
IF (W0000000000000000) GO TO 513	SIN	409
IF (W0000000000000000) GO TO 513	SIN	410
GO TO 515	SIN	411
375 SW=ON	SIN	412
IF (W0000000000000000) WRITE(15,2132)	SIN	413
C	SIN	414
375 SW=ON	SIN	415

GE/EE/72-7

```

C      PC=1
C
C      ***PREPARE TO OPERATE MODE
IF(SI=VAL.EC.1) WRITE(6,3025)
P(1)=P(2)=P(3)=1
PC=2
7(5)=7(4)=7(3)=7(2)=7(1)=2
S*(2)=3
S*(3)=1
C      J=1
C      E=8
C      O(2)=1
C      O(4)=2
C      O(5)=2
C      AS=OFF
C      J=1
C
C      COMMENT **SYNC IT COUNTER 1
IF(SIGVAL.EC.1) WRITE(6,3010)
C      O(1)=1
C
C      COMMENT **SYNC IT COUNTER 2
IF(SI=VAL.EC.1) WRITE(6,3015)
C      O(2)=1
C      DO 515 I=1,24
515 I(11)=I(20)
I(22)=I(24)=ONE
IF(REGISTER.EC.2) GO TO 520
REGISTER=REGISTER
CALL REG2
C      O(1)=1
C      PC=3
C
C      COMMENT **O(1) O(2) O(3) TOLD SUB-MODE OF NORMAL WALT
520 IF(SIGVAL.EC.1) WRITE(6,3020)
IF(SI(2).EC.2) K.E.FALT) GO TO 535
C      O(1)=1
C      GO TO 535
525 IF(SIGVAL.EC.1) WRITE(6,3025)
IF(SI(2).EC.2) K.E.FALT) GO TO 535
C      O(1)=1
C      GO TO 573
530 CONTINUE
C      O(2)=1
C      GO TO 533
C
C      COMMENT **INTERLOCK SUB-MODE OF NORMAL WALT
535 IF(SI(2).EC.2) K.E.FALT) GO TO 550
C      O(1)=1
C      IF(SIGVAL.EC.1) WRITE(6,3030)
540 IF(SI(2).EC.2) K.E.FALT) GO TO 545
IF(SI(2).EC.2) K.E.FALT) GO TO 555
C      O(2)=1
C      I*SIG=FSIG=1
C      GO TO 563
545 CONTINUE
C      PC=1
C      GO TO 625
550 IF(SIGVAL.EC.1) WRITE(6,3035)
C      O(1)=1
C      GO TO 525
555 FSIG=2
I*SIG=2
C      O(1)=1
C      GO TO 573
C
C      COMMENT ***PREPARE TO LOAD SUB-MODE OF NORMAL WALT
560 IF(SIGVAL.EC.1) WRITE(6,3040)
IF(SI(2).EC.2) K.E.FALT) GO TO 565
C      PC=1
C      GO TO 535
565 IF(SIGVAL.EC.1) WRITE(6,3045)
C      O(2)=1
C      FSIG=1
C      GO TO 45
C
C      COMMENT ***O(1) O(2) O(3) TOLD SUB-MODE OF NORMAL WALT
570 IF(SI(2).EC.2) K.E.FALT) GO TO 595
575 IF(SIGVAL.EC.1) WRITE(6,3045)
IF(SI(2).EC.2) K.E.FALT) GO TO 595
C      O(1)=1

```

SIN	415
SIN	416
SIN	417
SIN	418
SIN	419
SIN	420
SIN	421
SIN	422
SIN	423
SIN	424
SIN	425
SIN	426
SIN	427
SIN	428
SIN	429
SIN	430
SIN	431
SIN	432
SIN	433
SIN	434
SIN	435
SIN	436
SIN	437
SIN	438
SIN	439
SIN	440
SIN	441
SIN	442
SIN	443
SIN	444
SIN	445
SIN	446
SIN	447
SIN	448
SIN	449
SIN	450
SIN	451
SIN	452
SIN	453
SIN	454
SIN	455
SIN	456
SIN	457
SIN	458
SIN	459
SIN	460
SIN	461
SIN	462
SIN	463
SIN	464
SIN	465
SIN	466
SIN	467
SIN	468
SIN	469
SIN	470
SIN	471
SIN	472
SIN	473
SIN	474
SIN	475
SIN	476
SIN	477
SIN	478
SIN	479
SIN	480
SIN	481
SIN	482
SIN	483
SIN	484
SIN	485
SIN	486
SIN	487
SIN	488
SIN	489
SIN	490
SIN	491
SIN	492
SIN	493
SIN	494
SIN	495
SIN	496

```

IF(I1.EQ.0FF) GO TO 579
IF(SIGVAL.EQ.1) WRITE(6,3650)
FSSIG=2
GO TO 45
550 IF(SIGVAL.EQ.1) WRITE(6,3645)
C
  YF=1
  GO TO 635
555 CONTINUE
C
  D(2)=F
C
COMMENT **PREPARE TO COMPUTE SUB-PAGE OF MANUAL WALT
560 IF(SIGVAL.EQ.1) WRITE(6,3655)
C
  MD=1
C
  D=0
C
  K=1
  FSECT=NTIME=XSIG=XSING=FSSIG=I*SIG=TSIG=0
  GO TO 120
C
COMMENT **4BIT MODE
565 TSIG=1
  ST(2)=FSSIG=I*SIG=XSING=0
  FS=I=OFF
570 IF(SIGVAL.EQ.1) WRITE(6,3660)
575 CP(4)=CP(2)=CP(2)=CP(1)=ZERO
C
  CP(5)=1
  IF(I.EQ.0FF) GO TO 623
C
  RF=0
  IF(SF(2).EQ.1) GO TO 575
C
  D(4)=1
  GO TO 45
580 CONTINUE
C
  TC=1
  GO TO 639
585 WRITE(6,3665)
C
  TC=0
  TSIG=0
  GO TO 575
C
COMMENT **PREPARE TO SAMPLE MODE
590 IF(SIGVAL.EQ.1) WRITE(6,3670)
C
  J=0
C
COMMENT **SAMPLE CODE
  IF(SIGVAL.EQ.1) WRITE(6,3675)
C
  CP(1)=0
C
  Y(4)=1
  GO 635 I1=1,4
  IF(IT(I1).EQ.1) CP(I1)=ONE
595 CONTINUE
  SF(2)=IT(5)
  T=OFF
C
  PC=3
C
COMMENT **PARITY CHECK
  IF(SIGVAL.EQ.1) WRITE(6,3680)
C
  D(4)=1
  DO 645 I1=1,4
    IF(SF(I1).EQ.1) CP(I1)=ONE
  IF(CP(2).EQ.1) CP(1)=ONE
  GO TO 645
600 SP(3)=1
605 CONTINUE
C
  D(4)=0
C
  CP(5)=1
C
COMMENT **PROCESS CODE-GENERAL
  IF(CP(4).EQ.ZERO) GO TO 655
  CODE=L*CP(2)+2*CP(2)+CP(1)+1
  GO TO (57,675,515,63,555,715,775,785) CODE
C
COMMENT **PROCESS CODE-OCTAL
610 IF(SIGVAL.EQ.1) WRITE(6,3685)
C
  Y(4)=1
  DO 66. I2=1,2:
    L(25-I2)=L(22-I2)
  DO 655 I1=1,7
    L(I1)=CP(I1)
    IF(REGISTER.FQ.1) GO TO 785
    REGISTER=REGISTER(2)
    CALL F662
C
  D(4)=0
  GO TO 735
C

```

SIG	497
SIG	498
SIG	499
SIG	500
SIG	501
SIG	502
SIG	503
SIG	504
SIG	505
SIG	506
SIG	507
SIG	508
SIG	509
SIG	510
SIG	511
SIG	512
SIG	513
SIG	514
SIG	515
SIG	516
SIG	517
SIG	518
SIG	519
SIG	520
SIG	521
SIG	522
SIG	523
SIG	524
SIG	525
SIG	526
SIG	527
SIG	528
SIG	529
SIG	530
SIG	531
SIG	532
SIG	533
SIG	534
SIG	535
SIG	536
SIG	537
SIG	538
SIG	539
SIG	540
SIG	541
SIG	542
SIG	543
SIG	544
SIG	545
SIG	546
SIG	547
SIG	548
SIG	549
SIG	550
SIG	551
SIG	552
SIG	553
SIG	554
SIG	555
SIG	556
SIG	557
SIG	558
SIG	559
SIG	560
SIG	561
SIG	562
SIG	563
SIG	564
SIG	565
SIG	566
SIG	567
SIG	568
SIG	569
SIG	570
SIG	571
SIG	572
SIG	573
SIG	574
SIG	575
SIG	576
SIG	577
SIG	578
SIG	579


```

GO-TEXT      ** HILF2 READ
OAH=H244
STO=H250
CALL GML00C
IF(FLAG,SC,1: 50 TO 15

```

```

C.
COMPILE *EXECUTE
1025 PHASE=FSECT*TIME
      IF(CODE.4E.1.4D.CC.4E.9) GO TO 1030
      FSECT=ISECT+1
      GO TO 1035
1030 IF(CODE.4E.7) GO TO 1031
      IF(124.EC.ZERO) GO TO 1045
1031 FSECT=FSECT
      GO TO 1042
1035 IF(CASE.EC.FSECT) GO TO 1043
      FSECT=PHASE
1040 IF(FSECT.GT.123) FSECT=100(IFSECT,123)
      IF(1123.EC.ZERO) GO TO 1045
      IF(SEL.4L.EC.1) WRITE(4,4643)
      SECT=FSECT
      CALL FLASSTO
      IF(SEL.4L.EC.2) GO TO 15

```

2.45 cc TO (1755, 1756, 1757, 1758, 1759, 1760, 1761, 1762, 1763, 1764, 1765, 1766, 1767, 1768, 1769, 1770, 1771, 1772, 1773, 1774, 1775, 1776, 1777, 1778, 1779, 1780, 1781, 1782, 1783, 1784, 1785, 1786, 1787, 1788, 1789, 1790, 1791, 1792, 1793, 1794, 1795, 1796, 1797, 1798, 1799, 1800, 1801, 1802, 1803, 1804, 1805, 1806, 1807, 1808, 1809, 1810, 1811, 1812, 1813, 1814, 1815, 1816, 1817, 1818, 1819, 1820, 1821, 1822, 1823, 1824, 1825, 1826, 1827, 1828, 1829, 1830, 1831, 1832, 1833, 1834, 1835, 1836, 1837, 1838, 1839, 1840, 1841, 1842, 1843, 1844, 1845, 1846, 1847, 1848, 1849, 1850, 1851, 1852, 1853, 1854, 1855, 1856, 1857, 1858, 1859, 1860, 1861, 1862, 1863, 1864, 1865, 1866, 1867, 1868, 1869, 1870, 1871, 1872, 1873, 1874, 1875, 1876, 1877, 1878, 1879, 1880, 1881, 1882, 1883, 1884, 1885, 1886, 1887, 1888, 1889, 1890, 1891, 1892, 1893, 1894, 1895, 1896, 1897, 1898, 1899, 1900, 1901, 1902, 1903, 1904, 1905, 1906, 1907, 1908, 1909, 1910, 1911, 1912, 1913, 1914, 1915, 1916, 1917, 1918, 1919, 1920, 1921, 1922, 1923, 1924, 1925, 1926, 1927, 1928, 1929, 1930, 1931, 1932, 1933, 1934, 1935, 1936, 1937, 1938, 1939, 1940, 1941, 1942, 1943, 1944, 1945, 1946, 1947, 1948, 1949, 1950, 1951, 1952, 1953, 1954, 1955, 1956, 1957, 1958, 1959, 1960, 1961, 1962, 1963, 1964, 1965, 1966, 1967, 1968, 1969, 1970, 1971, 1972, 1973, 1974, 1975, 1976, 1977, 1978, 1979, 1980, 1981, 1982, 1983, 1984, 1985, 1986, 1987, 1988, 1989, 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023, 2024, 2025, 2026, 2027, 2028, 2029, 2030, 2031, 2032, 2033, 2034, 2035, 2036, 2037, 2038, 2039, 2040, 2041, 2042, 2043, 2044, 2045, 2046, 2047, 2048, 2049, 2050, 2051, 2052, 2053, 2054, 2055, 2056, 2057, 2058, 2059, 2060, 2061, 2062, 2063, 2064, 2065, 2066, 2067, 2068, 2069, 2070, 2071, 2072, 2073, 2074, 2075, 2076, 2077, 2078, 2079, 2080, 2081, 2082, 2083, 2084, 2085, 2086, 2087, 2088, 2089, 2090, 2091, 2092, 2093, 2094, 2095, 2096, 2097, 2098, 2099, 2100, 2101, 2102, 2103, 2104, 2105, 2106, 2107, 2108, 2109, 2110, 2111, 2112, 2113, 2114, 2115, 2116, 2117, 2118, 2119, 2120, 2121, 2122, 2123, 2124, 2125, 2126, 2127, 2128, 2129, 2130, 2131, 2132, 2133, 2134, 2135, 2136, 2137, 2138, 2139, 2140, 2141, 2142, 2143, 2144, 2145, 2146, 2147, 2148, 2149, 2150, 2151, 2152, 2153, 2154, 2155, 2156, 2157, 2158, 2159, 2160, 2161, 2162, 2163, 2164, 2165, 2166, 2167, 2168, 2169, 2170, 2171, 2172, 2173, 2174, 2175, 2176, 2177, 2178, 2179, 2180, 2181, 2182, 2183, 2184, 2185, 2186, 2187, 2188, 2189, 2190, 2191, 2192, 2193, 2194, 2195, 2196, 2197, 2198, 2199, 2200, 2201, 2202, 2203, 2204, 2205, 2206, 2207, 2208, 2209, 2210, 2211, 2212, 2213, 2214, 2215, 2216, 2217, 2218, 2219, 2220, 2221, 2222, 2223, 2224, 2225, 2226, 2227, 2228, 2229, 2230, 2231, 2232, 2233, 2234, 2235, 2236, 2237, 2238, 2239, 2240, 2241, 2242, 2243, 2244, 2245, 2246, 2247, 2248, 2249, 2250, 2251, 2252, 2253, 2254, 2255, 2256, 2257, 2258, 2259, 2260, 2261, 2262, 2263, 2264, 2265, 2266, 2267, 2268, 2269, 2270, 2271, 2272, 2273, 2274, 2275, 2276, 2277, 2278, 2279, 2280, 2281, 2282, 2283, 2284, 2285, 2286, 2287, 2288, 2289, 2290, 2291, 2292, 2293, 2294, 2295, 2296, 2297, 2298, 2299, 2300, 2301, 2302, 2303, 2304, 2305, 2306, 2307, 2308, 2309, 2310, 2311, 2312, 2313, 2314, 2315, 2316, 2317, 2318, 2319, 2320, 2321, 2322, 2323, 2324, 2325, 2326, 2327, 2328, 2329, 2330, 2331, 2332, 2333, 2334, 2335, 2336, 2337, 2338, 2339, 2340, 2341, 2342, 2343, 2344, 2345, 2346, 2347, 2348, 2349, 2350, 2351, 2352, 2353, 2354, 2355, 2356, 2357, 2358, 2359, 2360, 2361, 2362, 2363, 2364, 2365, 2366, 2367, 2368, 2369, 2370, 2371, 2372, 2373, 2374, 2375, 2376, 2377, 2378, 2379, 2380, 2381, 2382, 2383, 2384, 2385, 2386, 2387, 2388, 2389, 2390, 2391, 2392, 2393, 2394, 2395, 2396, 2397, 2398, 2399, 2400, 2401, 2402, 2403, 2404, 2405, 2406, 2407, 2408, 2409, 2410, 2411, 2412, 2413, 2414, 2415, 2416, 2417, 2418, 2419, 2420, 2421, 2422, 2423, 2424, 2425, 2426, 2427, 2428, 2429, 2430, 2431, 2432, 2433, 2434, 2435, 2

1455 SC TO (1245,129),1095,1195,1245,1295,1195,1535;1245,1195,1535,1295
1,1355,1425,1425,1495,1555,1235,1525,1335,1313,1335,1315,1445
2,1441,1595,1195,1433,1435,1435,1435) KCMAN

1555 SECT=16*22(5)+3*53(4)+4*25(3)+2*67(2)+C9(1)
IF(SCT=17.65.17) GO TO 1495
X=3
GO TO (1553, 1554, 1573, 1555, 1564, 1574, 1594, 1524, 1555, 1515, 1523, 1610
1, 1565, 1575, 1593, 1625) XCFAN

```

C
COMMENT      **INSTRUCTION SEARCH
1005  IC=IK+IS*CF(1)+3*CF(2)+4*CF(3)+2*CF(2)+CF(1)+1
      IF(1+IS,20,1) GO TO 1072
      IF(1(23)+59,2520) GO TO 1055
      CODE=**I(12)+4*I(15)+2*I(14)+I(13)+1
      IS=FC(ISECT-1,10)+1
      IF(CODE,20,12) ISECT=ISECT
      IF(CODE,31,12) ISECT=ISECT+CODE-11
      IF(CODE,47,12) ISECT=ISECT+16+CODE-11
      IF(ISECT,37,123) ISECT=37(ISECT,123)
      GO TO 1075
1005  ISECT=4+I(15)+32*I(17)+16*I(17)+5*I(16)+4*I(15)+2*I(14)+I(13)+1
      GO TO 1075
1072  ISECT=455CT
      TRANS=TRANS+3

```

```

C
COPYCT      **INSTRUCTION 2509
1079 CHAN=ICHAN
      SECT=ISECT
      IFEG=0
      CALL UMLDAD
      IF(FLAG.EC.1) GC TO 15
      IFEG=0
      GC 1,2- 11=1,24

```

1345 I(11)=4(I1)
IF(=CISTP,=3.0) GO TO 1525
P=CISTP=K75(3)
CALL P62

2:45 IF (X.E.SINGL) GO TO 1025
KEMALT
GO TO 725

C
 1:05 24175(15,4150)
 27175=1
 GO TO 1150
 11:5 IF(2561570.25.3) GO TO 1:60
 256157=2525(1)
 CALL 0562
 GO TO 1260
 12: IF(2561570.25.3) GO TO 1680
 A(12)=A(13)=75
 256157=2525(1)
 CALL 0562
 A(12)=A(13)=7520

SIN	745
SIN	746
SIN	747
SIN	748
SIN	749
SIN	750
SIN	751
SIN	752
SIN	753
SIN	754
SIN	755
SIN	756
SIN	757
SIN	758
SIN	759
SIN	760
SIN	761
SIN	762
SIN	763
SIN	764
SIN	765
SIN	766
SIN	767
SIN	768
SIN	769
SIN	770
SIN	771
SIN	772
SIN	773
SIN	774
SIN	775
SIN	776
SIN	777
SIN	778
SIN	779
SIN	780
SIN	781
SIN	782
SIN	783
SIN	784
SIN	785
SIN	786
SIN	787
SIN	788
SIN	789
SIN	790
SIN	791
SIN	792
SIN	793
SIN	794
SIN	795
SIN	796
SIN	797
SIN	798
SIN	799
SIN	800
SIN	801
SIN	802
SIN	803
SIN	804
SIN	805
SIN	806
SIN	807
SIN	808
SIN	809
SIN	810
SIN	811
SIN	812
SIN	813
SIN	814
SIN	815
SIN	816
SIN	817
SIN	818
SIN	819
SIN	820
SIN	821
SIN	822
SIN	823
SIN	824
SIN	825
SIN	826

Reproduced from
best available copy.

GO TO 1155	SIM	327
COMMENT **TRANSFER (IP2)	SIM	328
1115 IF(SIGNAL.EQ.1) WRITE(6,4322)	SIM	329
WRITE=FSSET=PMASE=6	SIM	330
IP2=1	SIM	331
IF(CHAN.EQ.27.AND.CHAN.EQ.29.AND.CHAN.EQ.30) GO TO 1120	SIM	332
WRITE(6,4323)	SIM	333
WRITE(6,4324)	SIM	334
GO TO 15	SIM	335
1120 DO 1125 I1=1,5	SIM	336
1125 CF(I1)=C(I1)	SIM	337
GO TO 1167	SIM	338
C	SIM	339
COMMENT **TRANSFER ON SIMUS (TMI)	SIM	340
1125 IF(SIGNAL.EQ.1) WRITE(6,4325)	SIM	341
IF(4(24).EQ.1) GO TO 1125	SIM	342
WRITE=1	SIM	343
GO TO 1167	SIM	344
C	SIM	345
COMMENT **CLEAR & ADD TO ACCUMULATOR (CLA)	SIM	346
1125 IF(SIGNAL.EQ.1) WRITE(6,4326)	SIM	347
DO 1145 I1=1,24	SIM	348
1145 A(I1)=Z(I1)	SIM	349
WRITE=1	SIM	350
GO TO 1125	SIM	351
C	SIM	352
COMMENT **ADD (ADD)	SIM	353
1145 IF(SIGNAL.EQ.1) WRITE(6,4327)	SIM	354
SUM=24	SIM	355
1150 I2=1	SIM	356
WRITE=1	SIM	357
1155 A=Z(I2)	SIM	358
DO 1160 I1=I2,SUM	SIM	359
CODE=4*(I1+2*N(I1))+K	SIM	360
IF(CODE.EQ.1.AND.4(I1).EQ.1) AK=CODE	SIM	361
IF(4(I1).EQ.2.AND.4(I1).EQ.2) AK=ZERO	SIM	362
A(I1)=Z(I1)	SIM	363
IF(CODE.EQ.1.CP.CODE.EQ.2.CP.CODE.EQ.4.CP.CODE.EQ.7) A(I1)=ONE	SIM	364
1160 CONTINUE	SIM	365
IF(I2.EQ.14) GO TO 1110	SIM	366
IF(SUM.EQ.24) GO TO 1155	SIM	367
I2=14	SIM	368
SUM=24	SIM	369
GO TO 1155	SIM	370
C	SIM	371
COMMENT **SUBTRACT (SUB)	SIM	372
1145 IF(SIGNAL.EQ.1) WRITE(6,4328)	SIM	373
SUM=24	SIM	374
1170 I2=1	SIM	375
WRITE=1	SIM	376
1175 A=Z(I2)	SIM	377
DO 1180 I1=I2,SUM	SIM	378
CODE=4*(I1+2*N(I1))+K	SIM	379
IF(CODE.EQ.1.AND.4(I1).EQ.2) AK=ONE	SIM	380
IF(4(I1).EQ.2.AND.4(I1).EQ.2) AK=ZERO	SIM	381
A(I1)=Z(I1)	SIM	382
IF(CODE.EQ.1.CP.CODE.EQ.2.CP.CODE.EQ.4.CP.CODE.EQ.7) A(I1)=ONE	SIM	383
1180 CONTINUE	SIM	384
IF(I2.EQ.14) GO TO 1110	SIM	385
IF(SUM.EQ.24) GO TO 1155	SIM	386
I2=14	SIM	387
SUM=24	SIM	388
GO TO 1175	SIM	389
C	SIM	390
COMMENT **SPLIT AND (SAND)	SIM	391
1195 IF(SIGNAL.EQ.1) WRITE(6,4329)	SIM	392
SUM=11	SIM	393
GO TO 1155	SIM	394
C	SIM	395
COMMENT **SPLIT SUBTRACT (SSUB)	SIM	396
1195 IF(SIGNAL.EQ.1) WRITE(6,4330)	SIM	397
SUM=11	SIM	398
GO TO 1175	SIM	399
C	SIM	400
COMMENT **MULTIPLY AND PROCEED (MPP)	SIM	401
1195 IF(SIGNAL.EQ.1) WRITE(6,4331)	SIM	402
WRITE=1	SIM	403
GO TO 795	SIM	404
C	SIM	405
COMMENT **CONSISTENT (CON)	SIM	406
1205 IF(SIGNAL.EQ.1) WRITE(6,4332)	SIM	407
	SIM	408
	SIM	409

```

MTIME=1
DO 1212 I1=1,24
IF(1(I1).EQ.ZERO) GO TO 1235
A(I1)=ZERO
GO TO 1212
1225 A(I1)=ONE
1210 CONTINUE
I1=ONE
DO 1215 I1=1,24
IF(1(I1)+A(I1).GT.2) GO TO 1228
1215 A(I1)=ZERO
1228 A(I1)=ONE
GO TO 1195

C
COMMENT **LEADS MAGNITUDE (VIM)
1235 IF(SIGNAL.EQ.1) WRITE(6,4139)
IF(1(24).EQ.3) GO TO 1210
WRITE=1
GO TO 1135

C
COMMENT **LOGICAL AND TO ACCUMULATOR (ANA)
1235 IF(SIGNAL.EQ.1) WRITE(6,4113)
WRITE=1
DO 1235 I1=1,24
1235 A(I1)=AND(A(I1),L(I1))
GO TO 1135

C
COMMENT **RESET DETECTOR (RSD)
1240 IF(SIGNAL.EQ.1) WRITE(6,4611)
WRITE=1
J=9
GO TO 1160

C
COMMENT **PRIMARY OUTPUT A (POA)
1245 IF(SIGNAL.EQ.1) WRITE(6,4112)
WRITE=1
1255 WRITE=1
IF(G(47).EQ.0) GO TO 1265
DO 1255 I1=1,24
IF(A(I1).DNE.EQ.ONE) GO TO 1250
1255 A(I1)=ZERO
1260 A(I1)=ONE
GO TO 1255
1255 I1=17
IF(A(I1)-ONE.EQ.ZERO) GO TO 1275
A(I1)=ONE
1275 I1=I1+1
IF(I1.EQ.25) GO TO 1232
IF(A(I1).EQ.ONE) GO TO 1275
A(I1)=ONE
GO TO 1275
1275 A(I1)=ZERO
1245 G(47)=0
IF(1(24).EQ.ONE) G(17)=1
IF(G(47).EQ.-1) WRITE(6,4343) STOP
IF(G(47).EQ.0) WRITE(6,4553) STOP
GO TO 1135

C
COMMENT **SECONDARY OUTPUT B (SOB)
1255 IF(SIGNAL.EQ.1) WRITE(6,4613)
WRITE=2
GO TO 1255

C
COMMENT **Tertiary Output C (TOC)
1255 IF(SIGNAL.EQ.1) WRITE(6,4614)
WRITE=3
GO TO 1255

C
COMMENT **QUATERNARY OUTPUT D (QOD)
1255 IF(SIGNAL.EQ.1) WRITE(6,4615)
WRITE=4
IF(22.EQ.CF) GO TO 1255
DO 1255 I1=1,5
1255 A(I1)=OR(A(I1),
((CF-15*(5-I1)*2*(4-I1)*3*(3-I1)*2*(2-I1)*2*(1-I1))/
(CF-CF-EQ.5) GO TO 1.63
IF(CODES.EQ.5.CP.CODES-2.4.CP.CODES.EQ.7) GO TO 1255
IF(CODES.EQ.11.AND.EQ.10.1.CP.CODES.EQ.21.AND.EQ.20.1) GO TO 1255
WRITE(6,4643) CODE
GO TO 1255
1262 CODE=4
WRITE(6,4643) CODE

```

S14	919
S10	911
S10	912
S12	913
S10	914
S10	915
S14	916
S14	917
S10	918
S14	919
S12	920
S10	921
S14	922
S14	923
S14	924
S10	925
S14	926
S12	927
S14	928
S12	929
S14	930
S10	931
S14	932
S10	933
S12	934
S14	935
S12	936
S10	937
S14	938
S12	939
S14	940
S14	941
S10	942
S10	943
S14	944
S10	945
S14	946
S14	947
S12	948
S10	949
S14	950
S10	951
S14	952
S10	953
S14	954
S10	955
S14	956
S14	957
S14	958
S10	959
S14	960
S10	961
S10	962
S10	963
S10	964
S14	965
S10	966
S14	967
S10	968
S14	969
S10	970
S14	971
S14	972
S10	973
S14	974
S14	975
S10	976
S14	977
S12	978
S12	979
S14	980
S10	981
S14	982
S12	983
S12	984
S12	985
S12	986
S10	987
S12	988
S10	989
S10	990
S14	991

```

      CC25=2*G(2)+D(1)
      WRITE(5,4243) C005
      GO TO 1353
1355 WRITE(5,4244)
      GO TO 1356
C
COMMENT **DISCRETE INPUT A (SIA)
1310 IF(SIGNAL.EQ.1) WRITE(5,4216)
      NTIME=1
      Y1=X1+1
      IF(X1.GT.X2) GO TO 1315
      GO TO 1322
1315 WRITE(6,4245)
      X1=Y1-1
1323 DO 1325 I1=1,19
1325 A(I1)=X(I1,X1)
      GO 1333 I1=1,5
1335 A(I1+10)=Z5R0
      IF(C0.EQ.1) A(23)=ONE
      IF(C0.EQ.1) A(21)=ONE
      IF(C0.EQ.1) A(23)=ONE
      IF(C0.EQ.1) A(23)=ONE
      IF(C0.EQ.1) A(24)=ONE
      GO TO 1175
C
COMMENT **DISCRETE INPUT B (SIB)
1335 IF(SIGNAL.EQ.1) WRITE(6,4217)
      NTIME=1
      Y1=Y1+1
      IF(Y1.GT.Y2) GO TO 1342
      GO TO 1345
1340 WRITE(6,4246)
      Y1=Y1-1
1345 DO 1359 I1=1,24
1359 A(I1)=Y(I1,Y1)
      GO TO 1175
C
COMMENT **VOLTAGE OUTPUT A (V0A)
1355 IF(SIGNAL.EQ.1) WRITE(6,4218)
      NTIME=1
      I2=3
      IF(I2.EQ.1) I2=16
      DO 1355 I1=1,3
1355 W(I1)=A(I1+I2)
      VOLTAGE=0.
      GO 1375 I1=1,3
      IF(W(I1).EQ.0) GO TO 1379
      VOLTAGE=VOLTAGE+W(I1)
      GO TO 1375
1375 VOLTAGE=VOLTAGE+W(I1)
1375 CONTINUE
      GO TO (1383,1385,1387) NCM
1381 WRITE(5,4251) (I2(I1-I1),I1=1,3),VOLTAGE
      GO TO 1396
1385 WRITE(6,4252) (W(I1-I1),I1=1,3),VOLTAGE
      GO TO 1396
1387 WRITE(5,4253) (W(I1-I1),I1=1,3),VOLTAGE
1395 PULSE=0.5*(2+2*P(2)+P(1))
      IF(PULSE.EQ.0) GO TO 1415,1417,1419,1421,1423,1425,1427,1429,1431,1433
      WRITE(5,4254)
      GO TO 1382
1402 WRITE(6,4255) NCM
      GO TO 1382
1405 WRITE(6,4256) NCM
      GO TO 1402
1415 WRITE(6,4257) NCM
      GO TO 1382
1415 WRITE(5,4258) NCM
      GO TO 1382
C
COMMENT **VOLTAGE OUTPUT B (V0B)
1420 IF(SIGNAL.EQ.1) WRITE(5,4219)
      NTIME=2
      GO TO 1352
C
COMMENT **VOLTAGE OUTPUT C (V0C)
1475 IF(SIGNAL.EQ.1) WRITE(6,4220)
      NTIME=2
      GO TO 1352
C
COMMENT **LOAD - USE REGISTER (L00)
1485 IF(SIGNAL.EQ.1) WRITE(6,4221)

```

```

SIN 992
SIN 993
SIN 994
SIN 995
SIN 996
SIN 997
SIN 998
SIN 999
SIN 1000
SIN 1001
SIN 1002
SIN 1003
SIN 1004
SIN 1005
SIN 1006
SIN 1007
SIN 1008
SIN 1009
SIN 1010
SIN 1011
SIN 1012
SIN 1013
SIN 1014
SIN 1015
SIN 1016
SIN 1017
SIN 1018
SIN 1019
SIN 1020
SIN 1021
SIN 1022
SIN 1023
SIN 1024
SIN 1025
SIN 1026
SIN 1027
SIN 1028
SIN 1029
SIN 1030
SIN 1031
SIN 1032
SIN 1033
SIN 1034
SIN 1035
SIN 1036
SIN 1037
SIN 1038
SIN 1039
SIN 1040
SIN 1041
SIN 1042
SIN 1043
SIN 1044
SIN 1045
SIN 1046
SIN 1047
SIN 1048
SIN 1049
SIN 1050
SIN 1051
SIN 1052
SIN 1053
SIN 1054
SIN 1055
SIN 1056
SIN 1057
SIN 1058
SIN 1059
SIN 1060
SIN 1061
SIN 1062
SIN 1063
SIN 1064
SIN 1065
SIN 1066
SIN 1067
SIN 1068
SIN 1069
SIN 1070
SIN 1071
SIN 1072
SIN 1073
SIN 1074

```

GE/EE/72-7

```

      NTIME=1
      DO 1425 I1=1,3
1425  P(I1)=3
      IF(C(I1).EQ.0X2) P(I1)=1
      IF(C(I2).EQ.0X2) P(I2)=1
      IF(C(I3).EQ.0X2) P(I3)=1
      WRITE(6,4122) (P(I1-I3), I1=1,3)
      GO TO 1463

C
COMMENT **ENTERED FINE COUNTDOWN (EFC)
1445 IF(SIGNAL.EC.1) WRITE(6,4122)
      NTIME=1
      FC=1
      GO TO 1457

C
COMMENT **HALT FINE COUNTDOWN (HFC)
1445 IF(SIGNAL.EC.1) WRITE(6,4123)
      NTIME=1
      FC=2
      GO TO 1463

C
COMMENT **SPLIT CONFAPES & LIMIT (ECL)
1455 IF(SIGNAL.EC.1) WRITE(6,4124)
      NTIME=2
      I2=9
1455 CODE=2
      PACE=3
      DO 1465 I1=1,19
1465  CODE=CODE+2**((I1-1)*1(I2+I1))
      IF(I2+I1).EQ.25=0) GO TO 1476
      DO 1475 I1=1,11
      IF(I1+I2).EQ.25=0) GO TO 1465
      I(I1+I2)=ZER0
      GO TO 1475
1475  I(I1+I2)=ONE
1475  CONTINUE
      DO 1475 I1=1,19
1475  PACE=PACE+2**((I1-1)*1(I2+I1))
      IF(PACE.LI.CODE) GO TO 1495
1476  DO 1495 I1=1,11
1476  I(I1+I2)=I(I1+I2)
1475  IF(I2.EQ.12) GO TO 1416
      I2=13
      GO TO 1455
1475  DO 1487 I1=1,11
1475  PACE=PACE+2**((I1-1)*1(I2+I1))
      IF(PACE.LE.CODE) GO TO 1476
      GO TO 1495

C
COMMENT **DIRECTOR CUTOUT (COB)
1485 IF(SIGNAL.EC.1) WRITE(6,4125)
      IF(SEC1.EQ.3) GO TO 1195
      NTIME=SEC1+1
      CODE=1+2(24)+4*1(22)+7*1(22)+1(21)
      IF(CODE.LI.7) GO TO 1585
      DO 1495 I1=1,5
      IF(CODE.EQ.12+7) GO TO 1518
1495  CONTINUE
1510 WRITE(6,4140) (1(25-I1), I1=1,4), CODE(I1)
      GO TO 1510
1515 WRITE(6,4147) (1(25-I1), I1=1,4), CODE
1510 SEC1=4
      GO TO 1521

C
COMMENT **ACCOMMODATE LEFT SHIFT (ALS)
1514 NTIME=1
1515 IF(SIGNAL.EC.1) WRITE(6,4125)
      NTIME=1
      IF(SEC1.LI.1) NTIME=2
      SEC1=SEC1+NTIME
      IF(SEC1.EQ.4) GO TO 1185
1525 DO 1533 I1=1,SEC1
      DO 1545 I2=1,23
1525  1(25-I2)=1(24-I2)
1535  1(1)=ZER0
      GO TO 1195

C
COMMENT **ACCOMMODATE RIGHT SHIFT (ARS)
1536 NTIME=1
1535 IF(SIGNAL.EC.1) WRITE(6,4127)
      NTIME=1
      IF(SEC1.LI.1) NTIME=2

```

SIP 1175
 SIP 1176
 SIP 1177
 SIP 1178
 SIP 1179
 SIP 1180
 SIP 1181
 SIP 1182
 SIP 1183
 SIP 1184
 SIP 1185
 SIP 1186
 SIP 1187
 SIP 1188
 SIP 1189
 SIP 1190
 SIP 1191
 SIP 1192
 SIP 1193
 SIP 1194
 SIP 1195
 SIP 1196
 SIP 1197
 SIP 1198
 SIP 1199
 SIP 1200
 SIP 1201
 SIP 1202
 SIP 1203
 SIP 1204
 SIP 1205
 SIP 1206
 SIP 1207
 SIP 1208
 SIP 1209
 SIP 1210
 SIP 1211
 SIP 1212
 SIP 1213
 SIP 1214
 SIP 1215
 SIP 1216
 SIP 1217
 SIP 1218
 SIP 1219
 SIP 1220
 SIP 1221
 SIP 1222
 SIP 1223
 SIP 1224
 SIP 1225
 SIP 1226
 SIP 1227
 SIP 1228
 SIP 1229
 SIP 1230
 SIP 1231
 SIP 1232
 SIP 1233
 SIP 1234
 SIP 1235
 SIP 1236
 SIP 1237
 SIP 1238
 SIP 1239
 SIP 1240
 SIP 1241
 SIP 1242
 SIP 1243
 SIP 1244
 SIP 1245
 SIP 1246
 SIP 1247
 SIP 1248
 SIP 1249
 SIP 1250
 SIP 1251
 SIP 1252
 SIP 1253
 SIP 1254
 SIP 1255
 SIP 1256

```

      SECT=SECT+NUM
      IF(SECT.EQ.0) GO TO 1105
      I2=A(24)
      DO 1545 I1=1,SECT
      1545 1545 I2=1,23
      1545 A(I2)=A(I2+1)
      1545 A(24)=I3
      GO TO 1135
C
C**SPLIT ACCUMULATOR LEFT SHIFT (SAL)
1549 NUM=1
1550 IF(SIGNAL.EQ.1) WRITE(6,4020)
      NTIME=SECT+1
      IF(SECT.LE.1) NTIME=2
      SECT=SECT+NUM
      IF(SECT.EQ.0) GO TO 1110
      DO 1555 I1=1,SECT
      1555 1555 I2=1,19
      A(25-I2)=A(24-I2)
      1555 A(12-I2)=A(11-I2)
      1555 A(14)=A(1)+2*0
      GO TO 1110
C
C**SPLIT ACCUMULATOR RIGHT SHIFT (SAR)
1554 NUM=1
1555 IF(SIGNAL.EQ.1) WRITE(6,4020)
      NTIME=SECT+1
      IF(SECT.LE.1) NTIME=2
      SECT=SECT+NUM
      IF(SECT.EQ.0) GO TO 1110
      I2=A(24)
      I4=A(12)
      DO 1575 I1=1,SECT
      1575 1575 I2=1,19
      A(I2)=A(I2+1)
      1575 A(12+I2)=A(11+I2)
      A(11)=I4
      1575 A(24)=I3
      GO TO 1110
C
C**SPLIT LEFT WORD LEFT SHIFT (SL)
1579 NUM=1
1580 IF(SIGNAL.EQ.1) WRITE(6,4030)
      NTIME=SECT+1
      IF(SECT.LE.1) NTIME=2
      SECT=SECT+NUM
      IF(SECT.EQ.0) GO TO 1110
      DO 1595 I1=1,SECT
      1595 1595 I2=1,19
      1595 A(25-I2)=A(24-I2)
      1595 A(14)=2*0
      GO TO 1110
C
C**SPLIT LEFT WORD RIGHT SHIFT (SLR)
1594 NUM=1
1595 IF(SIGNAL.EQ.1) WRITE(6,4030)
      NTIME=SECT+1
      IF(SECT.LE.1) NTIME=2
      SECT=SECT+NUM
      IF(SECT.EQ.0) GO TO 1110
      I2=A(24)
      DO 1605 I1=1,SECT
      1605 1605 I2=1,19
      A(12+I2)=A(11+I2)
      1605 A(24)=I3
      GO TO 1110
C
C**SPLIT RIGHT WORD LEFT SHIFT (SRL)
1609 NUM=1
1610 IF(SIGNAL.EQ.1) WRITE(6,4030)
      NTIME=SECT+1
      IF(SECT.LE.1) NTIME=2
      SECT=SECT+NUM
      IF(SECT.EQ.0) GO TO 1110
      DO 1625 I1=1,SECT
      1625 1625 I2=1,19
      A(12-I2)=A(11-I2)
      1625 A(1)=2*0
      GO TO 1110
C
C**SPLIT RIGHT WORD RIGHT SHIFT (SRR)
1624 NUM=1
1625 IF(SIGNAL.EQ.1) WRITE(6,4030)

```

```

SIR 1157
SIR 1158
SIR 1159
SIR 1160
SIR 1161
SIR 1162
SIR 1163
SIR 1164
SIR 1165
SIR 1166
SIR 1167
SIR 1168
SIR 1169
SIR 1170
SIR 1171
SIR 1172
SIR 1173
SIR 1174
SIR 1175
SIR 1176
SIR 1177
SIR 1178
SIR 1179
SIR 1180
SIR 1181
SIR 1182
SIR 1183
SIR 1184
SIR 1185
SIR 1186
SIR 1187
SIR 1188
SIR 1189
SIR 1190
SIR 1191
SIR 1192
SIR 1193
SIR 1194
SIR 1195
SIR 1196
SIR 1197
SIR 1198
SIR 1199
SIR 1200
SIR 1201
SIR 1202
SIR 1203
SIR 1204
SIR 1205
SIR 1206
SIR 1207
SIR 1208
SIR 1209
SIR 1210
SIR 1211
SIR 1212
SIR 1213
SIR 1214
SIR 1215
SIR 1216
SIR 1217
SIR 1218
SIR 1219
SIR 1220
SIR 1221
SIR 1222
SIR 1223
SIR 1224
SIR 1225
SIR 1226
SIR 1227
SIR 1228
SIR 1229
SIR 1230
SIR 1231
SIR 1232
SIR 1233
SIR 1234
SIR 1235
SIR 1236
SIR 1237
SIR 1238
SIR 1239

```

```

      NTIME=SECT+1
      IF (SECT.EQ.1) NTIME=2
      SECT=SECT+1
      IF (SECT.EQ.5) GO TO 1418
      I4=A(I1)
      DO 1625 I1=1,SECT
        TO 1622 I2=1,15
1626 A(I2)=A(I2+1)
1635 A(I1)=I4
      GO TO 1119
C
CONTINUE **MULTIPLY (P=1)
1540 IF (SIGMIL.EQ.1) WRITE(5,4324)
      NTIME=13
      I2=1
      I3=I6+23
      NTIME=73
      DO 1655 I1=1,24
1656 L(I1)=A(I1)
      IF (NTIME.EQ.5) GO TO 1655
      REGIST=REGS(2)
      CALL REG2
1655 CORREG(1)=CORREG(2)+ZER0
      I5=I3+1
      IF (I5.EQ.250) GO TO 1671
      DO 1665 I1=I2,I3
      IF (A(I1).EQ.ONE) GO TO 1668
      A(I1)=ONE
      GO TO 1665
1662 A(I1)=ZER0
1665 CONTINUE
1672 IF (I5.EQ.250) GO TO 1685
      DO 1695 I1=I2,I3
      IF (A(I1).EQ.ONE) GO TO 1675
      N(I1)=ONE
      GO TO 1693
1675 N(I1)=ZER0
1695 CONTINUE
1595 DO 1693 I1=1,15
      CORREG(1)=SHIFT(CORREG(1),1)
      CORREG(2)=SHIFT(CORREG(2),1)
      CORREG(3)=CORREG(1)+A(I5-I1)
1698 CORREG(1)=CORREG(2)+N(I5-I1)
      IF (I5.EQ.ONE) CORREG(1)=CORREG(1)+ONE
      IF (I5.EQ.ONE) CORREG(2)=CORREG(2)+ONE
      I4=CORREG(1)+CORREG(2)
      I4=SHIFT(I4,ONE)
      IF (I5.EQ.ONE) I4=N(I5).EQ.ONE; GO TO 1691
      IF (I5.EQ.250) I4=N(I5).EQ.ONE; GO TO 1692
      GO TO 1693
1691 A(I5)=ZER0
      GO TO 1694
1692 A(I5)=ONE
1697 IF (I5.EQ.ONE) I4=I4+ONE
1694 DO 1695 I1=I2,I3
      A(I1)=I4+I4+ONE
1695 I4=SHIFT(I4,-1)
      IF (I2.EQ.14) GO TO 1119
      IF (I2.EQ.23) GO TO 1115
      I2=14
      I2=23
      GO TO 1555
C
CONTINUE **MULTIPLY COEFFICD (P=1)
1701 IF (SIGMIL.EQ.1) WRITE(6,4325)
      DO 1705 I1=1,2
      IF (C(1).EQ.ZER0) C(1)=C(1).EQ.1 GO TO 1704
      IF (C(1).EQ.ONE) INT.P(I1).EQ.1 C(I1)=ZER0
      GO TO 1705
1704 C(I1)=ONE
1705 CONTINUE
      IF (C(1).EQ.1) GO TO 1713
      IF (C(2).EQ.250) INT.P(I2).EQ.1 GO TO 1709
      IF (C(2).EQ.ONE) INT.P(I2).EQ.1 C(I2)=ZER0
      GO TO 1715
1709 C(I2)=ONE
1710 C(I2)=5*C(5)+C(4)+C(3)+2*C(2)+C(1)+1
      SECT=SECT
      CALL UNLOAD
      IF (NPLAS.EQ.1) GO TO 15
      GO TO 1540
C

```

Reproduced from
best available copy.

SIN	1243
SIN	1241
SIN	1242
SIN	1243
SIN	1244
SIN	1245
SIN	1246
SIN	1247
SIN	1248
SIN	1249
SIN	1250
SIN	1251
SIN	1252
SIN	1253
SIN	1254
SIN	1255
SIN	1256
SIN	1257
SIN	1258
SIN	1259
SIN	1260
SIN	1261
SIN	1262
SIN	1263
SIN	1264
SIN	1265
SIN	1266
SIN	1267
SIN	1268
SIN	1269
SIN	1270
SIN	1271
SIN	1272
SIN	1273
SIN	1274
SIN	1275
SIN	1276
SIN	1277
SIN	1278
SIN	1279
SIN	1280
SIN	1281
SIN	1282
SIN	1283
SIN	1284
SIN	1285
SIN	1286
SIN	1287
SIN	1288
SIN	1289
SIN	1290
SIN	1291
SIN	1292
SIN	1293
SIN	1294
SIN	1295
SIN	1296
SIN	1297
SIN	1298
SIN	1299
SIN	1300
SIN	1301
SIN	1302
SIN	1303
SIN	1304
SIN	1305
SIN	1306
SIN	1307
SIN	1308
SIN	1309
SIN	1310
SIN	1311
SIN	1312
SIN	1313
SIN	1314
SIN	1315
SIN	1316
SIN	1317
SIN	1318
SIN	1319
SIN	1320
SIN	1321

COMMENT **SPLIT MULTIPLY (SNP)	SIN	1322
1715 IF(SIGNAL.EQ.1) WRITE(6,4925)	SIN	1323
NTIME=7	SIN	1324
I2=1	SIN	1325
I3=I6+19	SIN	1326
NTIME=10	SIN	1327
DO 1716 I1=1,11	SIN	1328
L(I1)=A(I1+12)	SIN	1329
1716 L(I1+13)=A(I1)	SIN	1330
L(1)=L(13)+ZERO	SIN	1331
IF(REGISTER.EQ.3) GO TO 1655	SIN	1332
REGISTER=REG(2)	SIN	1333
CALL RES2	SIN	1334
GO TO 1555	SIN	1335
C	SIN	1336
COMMENT **SPLIT MULTIPLY MODIFIED (SNP)	SIN	1337
1720 IF(SIGNAL.EQ.1) WRITE(6,4927)	SIN	1338
NG 1725 I1=1,2	SIN	1339
IF(C(I1).EQ.ZERO).AND.P(I1).EQ.1) GO TO 1724	SIN	1340
IF(C(I1).EQ.ZERO).AND.P(I1).EQ.1) C(I1)=ZERO	SIN	1341
GO TO 1725	SIN	1342
1724 C(I1)=ONE	SIN	1343
1725 CONTINUE	SIN	1344
IF(C(3).EQ.1) GO TO 1734	SIN	1345
IF(C(3).EQ.ZERO).AND.P(3).EQ.1) GO TO 1729	SIN	1346
IF(C(3).EQ.ZERO).AND.P(3).EQ.1) C(3)=ZERO	SIN	1347
GO TO 1732	SIN	1348
1729 C(3)=ONE	SIN	1349
1730 C(3)=15*C(5)+9*C(4)+4*C(2)+2*C(2)+C(1)+1	SIN	1350
SECT=SECT	SIN	1351
CALL UNLOAD	SIN	1352
IF(FLAG.EQ.1) GO TO 15	SIN	1353
GO TO 1725	SIN	1354
C	SIN	1355
COMMENT **STORE ACCUMULATOR (STO)	SIN	1356
1735 IF(SIGNAL.EQ.1) WRITE(6,4929)	SIN	1357
NTIME=1	SIN	1358
CHAM=CHAN	SIN	1359
SECT=SECT	SIN	1360
IF(C(1).EQ.22).AND.CHAM.EQ.23).AND.(CHAM.EQ.24).AND.(CHAM.EQ.25).AND.	SIN	1361
1 (CHAM.EQ.26) SECT=SECT-2	SIN	1362
IF(SECT.EQ.-1) SECT=127	SIN	1363
IF(SECT.EQ.0) SECT=128	SIN	1364
CALL STORE	SIN	1365
IF(FLAG.EQ.1) GO TO 15	SIN	1366
IF(C(3).EQ.4) GO TO 1135	SIN	1367
GO TO 1365	SIN	1368
C	SIN	1369
C	SIN	1370
C	SIN	1371
C	SIN	1372
C	SIN	1373
C	SIN	1374
C	SIN	1375
COMMENT **REACTING & TRANSLATION FORMAT STATEMENTS	SIN	1376
2300 FORMAT(//,2X,50(' '),/ ,2X, '---',52X, '---',/ ,3X, '---',12X, 'DATE COMPUTED')	SIN	1377
1 2301 '---',22X, '---',/ ,2X, '---',17X, 'SIMULATION PROGRAM',17X, '---',/ ,5X,	SIN	1378
2 2302 '---',52X, '---',/ ,2X, '---',4X, 'DATE= ',49,14X, 'TIME= ',19,4X, '---',	SIN	1379
3 2303 '---',52X, '---',/ ,3X,50(' '),/)	SIN	1380
2304 FORMAT(//,33X,55(' '),/ ,32X, '---',52X, '---',/ ,32X, '---',15X, '1174 COM')	SIN	1381
1 2305 'PUTER',24X, '---',/ ,32X, '---',17X, 'SIMULATION PROGRAM',17X, '---',/ ,	SIN	1382
2 2306 '31X, '---',52X, '---',/ ,32X, '---',4X, 'DATE= ',19,14X, 'TIME= ',19,4X,	SIN	1383
3 2307 '---',/ ,31X, '---',12X,2412,11X, '---',/ ,32X, '---',42X, '---',/ ,32X,56	SIN	1384
4 2308 (' '),/)	SIN	1385
2309 FORMAT(' TO RUN ANOTHER PROGRAM TYPE "RUN": TO STOP TYPE "WALT" -	SIN	1386
10)	SIN	1387
2310 FORMAT(13)	SIN	1388
2315 FORMAT(/ ,12X, '--- PRINTOUT OF INPUT PROGRAM ***' (ENTER PROGRAM	SIN	1389
13) //)	SIN	1390
2320 FORMAT(7A11,42)	SIN	1391
2321 FORMAT(14 ,7A19,42)	SIN	1392
2322 FORMAT(11,2A19)	SIN	1393
2323 FORMAT(' IF OUTPUT IS TO BE DISPOSED TO PRINTED, TYPE "P" AND "YOU	SIN	1394
19 NAME" OTHER-WISE TYPE "N" - 0)	SIN	1395
2325 FORMAT(// ,21X, '--- RESULTS OF SIMULATION ***')	SIN	1396
2335 FORMAT(72A11)	SIN	1397
2336 FORMAT(//,0 CONTINUE PROGRAM,/))	SIN	1398
2337 FORMAT(' THE FOLLOWING DATA IS NOT ALLOWED: ',72A11)	SIN	1399
2345 FORMAT(' LOAD CASES MUST BE IN RISEY WHEN DISNEY IS SPECIFIED')	SIN	1400
2350 FORMAT(' THE FOLLOWING INPUT DATA IS INVALID: ',72A11)	SIN	1401
2355 FORMAT(' COMPUTED IS NOT IN UNIT MODE - DATA CANNOT BE ENTERED - P	SIN	1402
2356 FORMAT(' TERMINATED')	SIN	1403

4320	FORMAT(° VOLTAGE OUTPUT °C INSTRUCTION - (VOC)°)	SIM	1487
4321	FORMAT(° LOAD PHASE REGISTER INSTRUCTION - (LFR)°)	SIM	1488
4322	FORMAT(° ENTER FINE COUNTDOWN INSTRUCTION - (EFC)°)	SIM	1489
4323	FORMAT(° HALT FINE COUNTDOWN INSTRUCTION - (HFC)°)	SIM	1490
4324	FORMAT(° SPLIT COMPARE & LIMIT INSTRUCTION - (SCL)°)	SIM	1491
4325	FORMAT(° CHARACTER OUTPUT INSTRUCTION - (COA)°)	SIM	1492
4326	FORMAT(° ACCUMULATOR LEFT SHIFT INSTRUCTION - (ALS)°)	SIM	1493
4327	FORMAT(° ACCUMULATOR RIGHT SHIFT INSTRUCTION - (ARS)°)	SIM	1494
4328	FORMAT(° SPLIT ACCUMULATOR LEFT SHIFT INSTRUCTION - (SAL)°)	SIM	1495
4329	FORMAT(° SPLIT ACCUMULATOR RIGHT SHIFT INSTRUCTION - (SAR)°)	SIM	1496
4330	FORMAT(° SPLIT LEFT WORD LEFT SHIFT INSTRUCTION - (SLU)°)	SIM	1497
4331	FORMAT(° SPLIT RIGHT WORD LEFT SHIFT INSTRUCTION - (SRL)°)	SIM	1498
4332	FORMAT(° SPLIT LEFT WORD RIGHT SHIFT INSTRUCTION - (SLR)°)	SIM	1499
4333	FORMAT(° SPLIT RIGHT WORD RIGHT SHIFT INSTRUCTION - (SRR)°)	SIM	1500
4334	FORMAT(° MULTIPLY INSTRUCTION - (MPY)°)	SIM	1501
4335	FORMAT(° MULTIPLY MODIFIED INSTRUCTION - (MPM)°)	SIM	1502
4336	FORMAT(° SPLIT MULTIPLY INSTRUCTION - (SMF)°)	SIM	1503
4337	FORMAT(° SPLIT MULTIPLY MODIFIED INSTRUCTION - (SMN)°)	SIM	1504
4338	FORMAT(° STOP ACCUMULATOR INSTRUCTION - (STO)°)	SIM	1505
4339	FORMAT(°, ° COMPUTE MODE°)	SIM	1506
4340	FORMAT(° FLAG STORE°)	SIM	1507
4341	FORMAT(14)	SIM	1508
4342	FORMAT(° PHASE REGISTER - P(3-1) = °, 211°)	SIM	1509
4343	FORMAT(° DISCRETE OUTPUT LINE 0°, 12°, °C HAS A "1" OUTPUT SIGNAL°)	SIM	1510
4344	FORMAT(° DISCRETE SWITCH IS OFF - DISCRETE OUTPUTS ARE DISABLED°)	SIM	1511
4345	FORMAT(° X-DISCRETE INPUTS USED HAVE EXCEEDED THOSE SUPPLIED - LAST 11 VALUES GIVEN WILL BE USED°)	SIM	1512
4346	FORMAT(° Y-DISCRETE INPUTS USED HAVE EXCEEDED THOSE SUPPLIED - LAST 11 VALUES GIVEN WILL BE USED°)	SIM	1513
4347	FORMAT(° BINARY CHARACTER OUTPUT - °, 401, 2X, °HEXDECIMAL CHARACTER 1 OUTPUT - °, 11°)	SIM	1514
4348	FORMAT(° BINARY CHARACTER OUTPUT - °, 401, 2X, °HEXDECIMAL CHARACTER 1 OUTPUT - °, 12°)	SIM	1515
4349	FORMAT(° BINARY OUTPUT ON LINE 0°, 11, °1 OF -1°)	SIM	1516
4350	FORMAT(° BINARY OUTPUT ON LINE 0°, 11, °0 OF -1°)	SIM	1517
4351	FORMAT(° V(13-1) = °, 301, ° WITH A VOLTAGE OUTPUT OF °, 7.2, ° VOLTS°)	SIM	1518
4352	FORMAT(° V(12-1) = °, 301, ° WITH A VOLTAGE OUTPUT OF °, 7.2, ° VOLTS°)	SIM	1519
4353	FORMAT(° V(13-1) = °, 301, ° WITH A VOLTAGE OUTPUT OF °, 7.2, ° VOLTS°)	SIM	1520
4354	FORMAT(° THE PHASE REGISTER IS IN THE IDEAL CONFIGURATION: AC VOLTA 15° OUTPUT°)	SIM	1521
4355	FORMAT(° VOLTAGE OUTPUT IS ON LINE 0°, 11, °3°)	SIM	1522
4356	FORMAT(° VOLTAGE OUTPUT IS ON LINE 0°, 11, °1°)	SIM	1523
4357	FORMAT(° VOLTAGE OUTPUT IS ON LINE 0°, 11, °2°)	SIM	1524
4358	FORMAT(° VOLTAGE OUTPUT IS ON LINE 0°, 11, °3°)	SIM	1525
4359	FORMAT(° A TRANSFER IS NOT ALLOWED TO L-REG, Y-, OR Z- LOCKS - FRO 157A° TERMINATED°)	SIM	1526
4360	FORMAT(° THE X-INSTRUCTION REQUESTED IS NOT AN INSTRUCTION°)	SIM	1527
END		SIM	1528
		SIM	1529
		SIM	1530
		SIM	1531
		SIM	1532
		SIM	1533
		SIM	1534

GE/EE/72-7

BLOCK DATA	SIN	1535
INTEGER CM, ZERO, ONE	SIN	1536
COMMON/410DATA/ CM, ZERO, ONE, KPEE(12), CM(27), MELDKE, MCOMPA,	SIN	1537
1 KLPAREN, KRPAREN, AIAES	SIN	1538
DATA MRES/1M1, 1M2, 1M3, 1M4, 1M5, 1M6, 1M7, 1M8, 1M9, 1M10	SIN	1539
DATA ZERO/12/, ONE/13/, NINES/9999999999/	SIN	1540
DATA M2/1M1, 1M2, 1M3, 1M4, 1M5, 1M6, 1M7, 1M8, 1M9, 1M10,	SIN	1541
1 1M11, 1M12, 1M13, 1M14, 1M15, 1M16, 1M17, 1M18, 1M19, 1M20,	SIN	1542
2 1M21, 1M22, 1M23, 1M24, 1M25, 1M26, 1M27, 1M28, 1M29, 1M30,	SIN	1543
DATA MCOMPA/1M1/, KLPAREN/1M1/, KRPAREN/1M1/, MELDKE/1M1 /	SIN	1544
DATA CM/2MCM/	SIN	1545
END	SIN	1546
	SIN	1547
	SIN	1548
	SIN	1549
	SIN	1550
	SIN	1551
	SIN	1552
	SIN	1553

SUBROUTINE STORE	SIM	1554
INTEGER A(24), CMX, CDE, CDEG(24,2), D, E(2), EM, F(4), FC	SIM	1555
INTEGER M(16), N, CDE, PHASE, R(24,4), REGIST, REGIST, SI, RK	SIM	1556
INTEGER SP(2), SECT, SIGNAL, U, V(24,4), VI, VK, X(13,13), XI	SIM	1557
INTEGER Y(24,13), YI, ZEP, DISPOSE	SIM	1558
COMMON A, CMX, CDE, CDEG, D, E, EM, F, FC, M, I(24), IPEG	SIM	1559
COMMON L(24), M(120,21), N(24), NCOL, NPLAS, NUP, NUP(172), PHASE	SIM	1560
COMMON SP, SIGNAL, P, REGIST, REGISTR, RI, RK, SECT, U, V, VI, VK	SIM	1561
COMMON X, XI, Y, YI, DISPOSE	SIM	1562
COMMON/INDATA/ CDE, ZERO, CDE, NREG(16), NREG(27), NPLANK, NCPMA,	SIM	1563
1 NLPAREN, NLPAREN, NLPAREN	SIM	1564
IF(CMX.EQ.21) GO TO 5	SIM	1565
IF(CMX.EQ.22.OR.CMX.EQ.23.OR.CMX.EQ.24) GO TO 1	SIM	1566
IF(CMX.EQ.25.OR.CMX.EQ.26.OR.CMX.EQ.27.OR.CMX.EQ.28) GO TO 19	SIM	1567
IF(CMX.EQ.29) GO TO 15	SIM	1568
IF(CMX.EQ.30) GO TO 35	SIM	1569
IF(EM.EQ.CD) GO TO 1	SIM	1570
WRITE(6,1-3)	SIM	1571
WRITE(4,1-3)	SIM	1572
NFLAG=1	SIM	1573
RETURN	SIM	1574
1 CALL L919	SIM	1575
RETURN	SIM	1576
5 IF(CD.EQ.0) GO TO 1	SIM	1577
WRITE(6,113)	SIM	1578
WRITE(4,113)	SIM	1579
NFLAG=1	SIM	1580
RETURN	SIM	1581
12 WRITE(6,125)	SIM	1582
WRITE(4,125)	SIM	1583
NFLAG=1	SIM	1584
RETURN	SIM	1585
15 IZ=CD(SECT-1,4)+1	SIM	1586
IF(FC.EQ.1) GO TO 25	SIM	1587
VK=ZERO	SIM	1588
DO 20 I1=1,24	SIM	1589
CDE=4*(I1-1)+2*(I1-1)+VK	SIM	1590
IF(V(I1,12).EQ.CDE.PHASE.A(I1).EQ.CDE) VK=ONE	SIM	1591
IF(V(I1,12).EQ.ZEP.A(I1).EQ.ZEP) VK=ZERO	SIM	1592
V(I1,12)=ZEP	SIM	1593
IF(CDE.EQ.1.OR.CDE.EQ.2.OR.CDE.EQ.4.OR.CDE.EQ.7) V(I1,12)=CDE	SIM	1594
20 CONTINUE	SIM	1595
21 IF(REGIST.EQ.5) RETURN	SIM	1596
REGIST=NREG(3)	SIM	1597
CALL REG2	SIM	1598
RETURN	SIM	1599
25 DO 30 I1=1,24	SIM	1600
V(I1,12)=A(I1)	SIM	1601
GO TO 21	SIM	1602
35 IF(FC.EQ.1) GO TO 55	SIM	1603
I4=CD(SECT-1,3)+1	SIM	1604
I2=1	SIM	1605
I3=11	SIM	1606
40 RK=ZERO	SIM	1607
DO 45 I1=I2,I3	SIM	1608
CDE=4*(I1-1)+2*(I1-1)+RK	SIM	1609
IF(R(I1,I4).EQ.CDE.PHASE.A(I1).EQ.CDE) RK=ONE	SIM	1610
IF(R(I1,I4).EQ.ZEP.A(I1).EQ.ZEP) RK=ZERO	SIM	1611
R(I1,I4)=ZEP	SIM	1612
IF(CDE.EQ.1.OR.CDE.EQ.2.OR.CDE.EQ.4.OR.CDE.EQ.7) R(I1,I4)=CDE	SIM	1613
45 CONTINUE	SIM	1614
IF(I3.EQ.24) GO TO 50	SIM	1615
I2=I4	SIM	1616
I3=24	SIM	1617
GO TO 40	SIM	1618
50 IF(REGISTR.EQ.3) RETURN	SIM	1619
REGIST=NREG(15)	SIM	1620
CALL REG2	SIM	1621
RETURN	SIM	1622
1-9 FORMAT(STORAGE CANNOT TAKE PLACE IN COLD STORAGE CHANNELS IF COL	SIM	1623
10-STOPAGE WRITE SWITCH IS OFF - PROGRAM TERMINATED)	SIM	1624
15 FORMAT(STORAGE IS NOT ALLOWED IN SINGLE-LOOPS (A,I,L, C, U) - PR	SIM	1625
16-STOPAGE TERMINATED)	SIM	1626
110 FORMAT(STORAGE CANNOT TAKE PLACE IN CHANNEL 50 IF DISCRETE SWIT	SIM	1627
12-STOPAGE IS OFF - PROGRAM TERMINATED)	SIM	1628
END	SIM	1629

SUBROUTINE L910	SIN	1630
INTEGER I(24), CMEM, CODE, COMREG(24,2), DB, E(8), EN, F(4), FC	SIN	1631
INTEGER M(16), CS, CME, PHASE, R(24,4), REGIST, REGISTR, FI, PK	SIN	1632
INTEGER S(2), SECT, SIGNAL, L, Y(24,4), VI, VK, X(19,12), XI	SIN	1633
INTEGER Y(24,12), YI, ZEPG, DISPOSE	SIN	1634
COMMON A, CMEM, CODE, COMREG, DB, E, EN, F, FC, M, I(24), IREG	SIN	1635
COMMON L(24), M(128,21), N(24), NCCL, NFLAG, NUP, NUPD(72), PHASE	SIN	1636
COMMON S2, SIGNAL, R, REGIST, REGISTR, RI, RK, SECT, U, V, VI, VK	SIN	1637
COMMON X, XI, Y, YI, DISPOSE	SIN	1638
COMMON/STPAT2/ CM, ZEPG, CME, IREG(18), NM(27), NCLANK, NCOMPA,	SIN	1639
1 ALFAREN, MFAREN, NIKES	SIN	1640
COMREG(11)=ZEPG	SIN	1641
DO 1 I1=1,24	SIN	1642
I2=SHIFT(COMREG(1),1)	SIN	1643
1 COMREG(11)=CR(I2,A(25-I1))	SIN	1644
IF(CM2.EC.21) GO TO 12	SIN	1645
IF(CM2.EC.22) GO TO 15	SIN	1646
IF(CM2.EC.23) GO TO 20	SIN	1647
IF(CM2.EC.24) GO TO 25	SIN	1648
IF(CM2.EC.25) GO TO 30	SIN	1649
IF(CM2.EC.26) GO TO 5	SIN	1650
WRITE(5,159)	SIN	1651
WRITE(4,159)	SIN	1652
NFLAG=1	SIN	1653
RETURN	SIN	1654
5 N(SECT,CM2)=COMREG(11)	SIN	1655
RETURN	SIN	1656
12 IF(DD.EC.6X) GO TO 5	SIN	1657
WRITE(5,159)	SIN	1658
WRITE(4,159)	SIN	1659
NFLAG=1	SIN	1660
RETURN	SIN	1661
15 I2=MOD(SECT-1,4)+1	SIN	1662
F(I2)=COMREG(1)	SIN	1663
REGIST=REG(6)	SIN	1664
GO TO 35	SIN	1665
20 I2=MOD(SECT-1,16)+1	SIN	1666
M(I2)=COMREG(1)	SIN	1667
REGIST=REG(4)	SIN	1668
GO TO 35	SIN	1669
25 I2=MOD(SECT-1,3)+1	SIN	1670
E(I2)=COMREG(1)	SIN	1671
REGIST=REG(7)	SIN	1672
GO TO 35	SIN	1673
30 U=COMREG(1)	SIN	1674
REGIST=REG(5)	SIN	1675
35 IF(REGIST.EC.8) RETURN	SIN	1676
CALL REG2	SIN	1677
RETURN	SIN	1678
2 3 FORMAT(10X) COLD STORAGE MEMORY CANNOT BE LOADED IF COLD-STORAGE UNIT	SIN	1679
15 SWITCH IS OFF - PROGRAM TERMINATED	SIN	1680
115 FORMAT(10X) HOT STORAGE MEMORY CANNOT BE LOADED IF DISCRETE SWITCH IS	SIN	1681
1 OFF - PROGRAM TERMINATED	SIN	1682
END	SIN	1683

SUBROUTINE UNLOAD	SIM	1684
INTEGER A(24), CHAN, CODE, COMRES(24,2), DD, E(3), EN, F(4), FC	SIM	1685
INTEGER H(18), C4, C4E, PHASE, P(24,4), REGIST, REGISTR, RI, RK	SIM	1686
INTEGER S3(3), SECT, SIGNAL, U, V(24,4), VI, VK, X(19,19), XI	SIM	1687
INTEGER Y(24,19), YI, Z(20, DISPOSE	SIM	1688
COMMON A, CHAN, CODE, COMRES, DD, E, EN, F, FC, H, I(24), IREG	SIM	1689
COMMON L(24), P(126,21), P(24), PCOL, PFLAG, RUM, RLOF(72), PHASE	SIM	1690
COMMON S3, SIGNAL, S, REGISTR, REGISTR, RI, RK, SECT, U, V, VI, VK	SIM	1691
COMMON X, XI, Y, YI, DISPOSE	SIM	1692
COMMON/4000/ ON, Z(20, CNE, KRES(18), NV(27), NPLANK, NCOMP2,	SIM	1693
1 ALAPEN, REFAREN, KIAES	SIM	1694
NUM=0	SIM	1695
DO 1 I1=1,11	SIM	1696
IF(CHAN.EQ.(21+I1)) GO TO (5,15,15,23,25,35,45,55,65,75,75) I1	SIM	1697
1 CONTINUE	SIM	1698
I2=MOD(SECT,CHAN)	SIM	1699
GO TO 37	SIM	1700
5 I2=MOD(SECT-1,4)+1	SIM	1701
I2=F(I2)	SIM	1702
GO TO 37	SIM	1703
10 I2=MOD(SECT-1,15)+1	SIM	1704
I2=H(I2)	SIM	1705
GO TO 37	SIM	1706
15 I2=MOD(SECT-1,3)+1	SIM	1707
I2=F(I2)	SIM	1708
GO TO 37	SIM	1709
20 I2=U	SIM	1710
GO TO 37	SIM	1711
25 DO 35 I1=1,24	SIM	1712
30 H(I1)=A(I1)	SIM	1713
GO TO 110	SIM	1714
25 DO 45 I1=1,24	SIM	1715
40 H(I1)=L(I1)	SIM	1716
45 GO TO 110	SIM	1717
50 IF(VI.EQ.1) NUM=1	SIM	1718
I2=MOD(SECT-1,4)+1	SIM	1719
DO 55 I1=1,24	SIM	1720
55 H(I1)=V(I1,I2)	SIM	1721
GO TO 90	SIM	1722
60 IF(PK.EQ.1) NUM=1	SIM	1723
I2=MOD(SECT-1,4)+1	SIM	1724
DO 65 I1=1,24	SIM	1725
65 H(I1)=K(I1,I2)	SIM	1726
GO TO 90	SIM	1727
70 I2=MOD(SECT-1,15)+1	SIM	1728
I2=MOD(I2+3,15)	SIM	1729
I2=H(I2)	SIM	1730
GO TO 90	SIM	1731
75 I2=MOD(SECT-1,3)+1	SIM	1732
I2=MOD(I2+4,3)	SIM	1733
I2=F(I2)	SIM	1734
80 IF(I2.EQ.NINES) GO TO 115	SIM	1735
DO 85 I1=1,24	SIM	1736
H(I1)=A(I1,I2,ONES)	SIM	1737
85 I2=SHIFT(I2,-1)	SIM	1738
90 IF(MUM.EQ.1) GO TO 95	SIM	1739
GO TO 115	SIM	1740
95 DO 105 I1=1,24	SIM	1741
IF(H(I1).EQ.ONES) GO TO 105	SIM	1742
H(I1)=ONE	SIM	1743
GO TO 105	SIM	1744
100 H(I1)=ZERO	SIM	1745
105 CONTINUE	SIM	1746
110 IF(IREG.EQ.1) RETURN	SIM	1747
IF(REGIST.EQ.2) RETURN	SIM	1748
REGIST=REG(4)	SIM	1749
CALL MSG2	SIM	1750
RETURN	SIM	1751
115 IF(IIFC.EQ.1) GO TO 125	SIM	1752
WRITE(5,175)	SIM	1753
WRITE(4,175)	SIM	1754
120 WRITE(1	SIM	1755
RETURN	SIM	1756
125 WRITE(6,175)	SIM	1757
WRITE(4,175)	SIM	1758
GO TO 120	SIM	1759
130 FORMAT(' OPERAND ADDRESS IS OUT OF RANGE - PROGRAM TERMINATED')	SIM	1760
135 FORMAT(' INSTRUCTION ADDRESS IS OUT OF RANGE - PROGRAM TERMINATED')	SIM	1761
1)	SIM	1762
END	SIM	1763

SUBROUTINE FLAGSTC	SIP	1764
INTEGER M(24), CMAN, CODE, CONFES(24,2), DO, E(3), EN, F(4), FC	SIN	1765
INTEGER M(15), OK, CNE, PHASE, R(24,4), REGIST, REGISTR, RI, RK	SIN	1766
INTEGER SE(2), SECT, SIGNAL, U, V(24,6), VI, VK, X(19,12), XI	SIN	1767
INTEGER Y(24,13), YI, ZERO, DISPOSE	SIN	1768
COMMON A, CMAN, CODE, CONFES, EN, E, EN, F, FC, N, I(24), IREG	SIN	1769
COMMON L(24), M(120,21), N(24), ACOL, AFLAG, KUN, NMO=0(72), PHASE	SIN	1770
COMMON SE, SIGNAL, P, REGIST, REGISTR, RI, RK, SECT, U, V, VI, VK	SIN	1771
COMMON X, XI, Y, YI, DISPOSE	SIN	1772
COMMON/140DAT/ ON, ZERO, CNE, NREG(12), N(27), NELANK, NCORNA,	SIN	1773
1 NLFABEN, NPAREN, NIDES	SIN	1774
MC=4*S2(2)+2*S3(2)+S2(1)+1	SIN	1775
IF(CODE.NE.12) GO TO 2	SIN	1776
IF(CMAN.NE.21.AND.CMAN.NE.22.AND.CMAN.NE.23.AND.CMAN.NE.24) GO TO 2	SIN	1777
IF(MO=57.6) GO TO 25	SIN	1778
WRITE(6,115)	SIN	1779
WRITE(4,115)	SIN	1780
NFLAG=1	SIN	1781
RETURN	SIN	1782
2 GO TO (1,5,13,15,26,25,25,46) KUN	SIN	1783
1 IF(SIGNAL.EQ.2.AND.PEDEST.EQ.1) WRITE(6,108)	SIN	1784
RETURN	SIN	1785
5 CMAN=22	SIN	1786
GO TO 45	SIN	1787
10 WRITE(6,115) (I(25-I1), I1=1,24)	SIN	1788
RETURN	SIN	1789
15 CMAN=21	SIN	1790
IF(SIGNAL.EQ.1.AND.PEDEST.EQ.1) WRITE(6,119)	SIN	1791
GO TO 45	SIN	1792
20 CMAN=24	SIN	1793
GO TO 45	SIN	1794
25 DO 35 I1=1,24	SIN	1795
30 L(I1)=A(I1)	SIN	1796
IF(PEDEST.EQ.3) RETURN	SIN	1797
REGIST=REG(2)	SIN	1798
CALL REG2	SIN	1799
RETURN	SIN	1800
25 CMAN=23	SIN	1801
GO TO 45	SIN	1802
40 CMAN=25	SIN	1803
45 CALL LOG3	SIN	1804
RETURN	SIN	1805
100 FORMAT(' FLAGGED INSTRUCTION STOP CODE - 1000')	SIN	1806
115 FORMAT(' FLAGGED INSTRUCTION TELEMETRY SIGNAL - 0,4(602,122)')	SIN	1807
130 FORMAT(' FLAGGED INSTRUCTION STORE IN TWO STORAGE CHANNELS')	SIN	1808
145 FORMAT(' FLAGSTORE IS ALLOWED ONLY IN L-250 WHEN STORE INSTRUCTION')	SIN	1809
1 IS TO CMAN SE, F, H, O E-LOOPS - (EXCEPT TERMINATED)')	SIN	1810
END	SIN	1811

```

SUBROUTINE DISPLAY
INTEGER I(24), CMX, CODE, CORREG(24,2), DO, E(3), EN, F(6), FC
INTEGER H(16), G, C, P, PHASE, S(24,4), REGIST, REGIST, RI, R,
INTEGER SP(7), SECT, SIGNAL, U, V(24,4), VI, VK, X(19,19), XI
INTEGER Y(24,19), YI, ZERO, Z(12), ZDISPSE
COMMON A, CMX, CODE, CORREG, CO, E, EN, F, FC, H, I(24), INES
COMMON L(24), M(128,24), M(24), MCOL, MFLAG, MUX, MUXC(72), PHASE
COMMON S, SIGNAL, R, REGIST, REGIST, RI, R, SECT, U, V, VI, VK
COMMON X, XI, Y, YI, ZDISPSE
COMMON/MPDATA/ ON, ZFC, CNE, MREG(18), MX(27), MFLANK, MCOMPA,
1 MFLAREX, MFLAREN, MFLRES
ENTRY REG2
DO 1 I1=1,16
IF(CORREG(I1,20,1)) GO TO 5
1 CONTINUE
RETURN
5 IF(REG(I1,20,1)) GO TO(10,15,20,25,30,40,50,60,70,75) I1
RETURN
10 WRITE(5,112) (I(25-I1),I1=1,24)
RETURN
15 WRITE(5,115) (L(25-I1),I1=1,24)
RETURN
20 WRITE(5,120) (Y(25-I1),I1=1,24)
RETURN
25 WRITE(5,125) (YI(25-I1),I1=1,24)
RETURN
30 I2=0
DO 35 I1=1,24
CORREG(I1)=MX(I2,CNE)
35 I2=SHIFT(I2,-1)
WRITE(5,130) (CORREG(25-I1),I1=1,24)
RETURN
40 I2=CO(SECT-1,4)+1
I3=F(I2)
DO 45 I1=1,24
CORREG(I1)=MX(I3,CNE)
45 I2=SHIFT(I3,-1)
I3=I2-1
WRITE(5,135) I3, (CORREG(25-I1),I1=1,24)
RETURN
50 I2=CO(SECT-1,3)+1
I3=E(I2)
DO 55 I1=1,24
CORREG(I1)=MX(I3,CNE)
55 I2=SHIFT(I3,-1)
I3=I2-1
WRITE(5,140) I3, (CORREG(25-I1),I1=1,24)
RETURN
60 I2=CO(SECT-1,15)+1
I3=M(I2)
DO 65 I1=1,24
CORREG(I1)=MX(I3,CNE)
65 I2=SHIFT(I3,-1)
I3=I2-1
WRITE(5,145) I3, (CORREG(25-I1),I1=1,24)
RETURN
70 I2=CO(SECT-1,4)+1
I3=I2-1
WRITE(5,150) I3, (V(25-I1,I2),I1=1,24)
RETURN
75 I2=CO(SECT-1,4)+1
I3=I2-1
WRITE(5,155) I3, (V(25-I1,I2),I1=1,24)
RETURN
ENTRY REG1
DO 80 I1=1,13
REG(I1)=0
REGIST=0
DO 85 I1=1,72
IF(MREG(I1,20,1)) GO TO 90
85 CONTINUE
WRITE(5,160) (MREG(I1),I1=1,72)
WRITE(5,165) (MREG(I1),I1=1,72)
MCOL=I1
RETURN
90 MCOL=I1
95 MCOL=MCOL+1
IF(MCOL.GT.72) RETURN
IF(MREG(MCOL,20,1).EQ.0) GO TO 100
IF(MREG(MCOL,20,1).EQ.1) RETURN
DO 105 I1=1,16
IF(MREG(MCOL,20,1)) GO TO 105

```

```

120 EC=TIME
    WRITE(5,164) MWD2(MCOL)
    WRITE(4,165) MWD3(MCOL)
    GO TO 95
125 REG(12)=1
    RFGIST=1
    GO TO 95
130 FC=27(0) 2(24-1) = 0,3(201,1X)
135 FC=27(0) 1(24-1) = 0,3(201,2X)
140 FC=27(0) 2(24-1) = 0,3(201,2X)
145 FC=27(0) 1(24-1) = 0,3(201,1X)
150 FC=27(0) 0(24-1) = 0,3(201,1X)
155 FC=27(0) 1(0,11,0,24-1) = 0,3(201,1X)
160 FC=27(0) 0(0,11,0,24-1) = 0,3(201,1X)
165 FC=27(0) 0(0,11,0,24-1) = 0,3(201,1X)
170 FC=27(0) 0(0,11,0,24-1) = 0,3(201,1X)
175 FC=27(0) 0(0,11,0,24-1) = 0,3(201,1X)
180 FC=27(0) 0(0,11,0,24-1) = 0,3(201,1X)
185 FC=27(14,1,0) IS NOT A VALID REGISTER DISPLAY ARGUMENT.
    EN

```

S12	1996
S1P	1935
S1A	1896
S14	1897
S1P	1898
S1E	1899
S14	1900
S14	1971
S1A	1902
S14	1903
S14	1934
S1P	1905
S1E	1906
S14	1907
S1E	1953
S1A	1939
S14	1910
S1A	1911
S1A	1912
S1E	1913

SUBROUTINE MEMOBY	SIM	1914
INTEGER A(24), CMAN, CODE, COMRES(24,2), DO, E(4), EN, F(4), FC	SIM	1915
INTEGER H(16), CK, DIF, PHASE, P(24,4), REGIST, REGISTQ, RI, SK	SIM	1916
INTEGER S(2), SECT, SIGNAL, U, V(24,4), VI, VK, X(19,13), XI	SIM	1917
INTEGER Y(24,13), YI, ZERG, DISPOSE	SIM	1918
COMMON A, CMAN, CODE, COMRES, DO, E, EN, F, FC, H, I(24), IFCG	SIM	1919
COMMON L(24), M(24,2), N(24), NCCL, NPLAS, NUP, NMODS(72), PHASE	SIM	1920
COMMON S2, SIGNAL, S, REGIST, REGISTQ, RI, RK, SECT, U, V, VI, VK	SIM	1921
COMMON X, XI, Y, YI, DISPOSE	SIM	1922
COMMON/INDATA/ CM, ZERO, CKE, KREG(15), M(27), NPLANK, NCPHRA,	SIM	1923
1 NLFACN, KOPACN, NIKES	SIM	1924
WRITE(5,163)	SIM	1925
NUL=ZERO	SIM	1926
1 NCCL=NCCL+1	SIM	1927
IF(NCCL.EQ.72) GO TO 55	SIM	1928
IF(MOD(NCCL),EQ.NPLAS) GO TO 5	SIM	1929
IF(MOD(NCCL),EQ.NPLANK) GO TO 55	SIM	1930
GO TO 1	SIM	1931
5 IF(MOD(NCCL+1),EQ.4*(21)) GO TO 15	SIM	1932
15 NCCL=NCCL+1	SIM	1933
IF(NCCL.EQ.72) GO TO 23	SIM	1934
IF(MOD(NCCL),EQ.NPLANK) GO TO 28	SIM	1935
IF(MOD(NCCL),EQ.NPLAS) GO TO 28	SIM	1936
GO TO 17	SIM	1937
25 NCCL=NCCL+1	SIM	1938
IF(NCCL.EQ.72) GO TO 55	SIM	1939
IF(MOD(NCCL),EQ.NPLANK) GO TO 55	SIM	1940
IF(MOD(NCCL),EQ.NPLAS) GO TO 55	SIM	1941
GO TO 15	SIM	1942
27 IF(DISPOSE.EQ.VK(13)) GO TO 87	SIM	1943
DO 55 I3=1,21	SIM	1944
CODE=ZERO	SIM	1945
DO 45 I2=1,124,2	SIM	1946
IF(X(I2,I3).EQ.VI*ES.AND.(I2+1,I3).EQ.NIKES) GO TO 45	SIM	1947
GO 40 I1=1,2	SIM	1948
PHASE=Y(I1+I2-1,I3)	SIM	1949
IF(PHASE.EQ.NIKES) GO TO 29	SIM	1950
GO 25 I4=1,24	SIM	1951
COMRES(I4,I1)=ATN(PHASE,CKE)	SIM	1952
25 PHASE=5*ATN(CHASE,-1)	SIM	1953
GO TO 45	SIM	1954
20 DO 25 I4=1,24	SIM	1955
35 COMRES(I4,I1)=77777777	SIM	1956
45 CONTINUE	SIM	1957
X(I1+I2-1,5) NUL, CODE, (COMRES(25-I4,I1),I4=1,24),I1=1,2)	SIM	1958
45 CODE=CODE+323	SIM	1959
55 NUL=NUL+323	SIM	1960
X(I1+I2-1,6,112)	SIM	1961
RETURN	SIM	1962
55 IF(PHASE.EQ.VK(13)) GO TO 30	SIM	1963
DO 75 I3=1,21	SIM	1964
CODE=ZERO	SIM	1965
DO 70 I2=1,124,4	SIM	1966
IF(X(I2,I3).EQ.VI*ES.AND.(I2+1,I3).EQ.NIKES.AND.(I2+2,I3).EQ.	SIM	1967
1 NIKES.AND.(I2+3,I3).EQ.NIKES) GO TO 75	SIM	1968
DO 65 I1=1,4	SIM	1969
IF(X(I1+I2-1,I3).EQ.NIKES) GO TO 69	SIM	1970
COMRES(I1)=Y(I1+I2-1,I3)	SIM	1971
GO TO 65	SIM	1972
65 COMRES(I1)=77777777	SIM	1973
65 CONTINUE	SIM	1974
Y(I1+I2-1,5) NUL, CODE, (COMRES(25-I4,I1),I4=1,24)	SIM	1975
70 CODE=CODE+343	SIM	1976
75 NUL=NUL+323	SIM	1977
WRITE(5,118)	SIM	1978
RETURN	SIM	1979
30 DO 87 I3=1,21	SIM	1980
CODE=ZERO	SIM	1981
DO 45 I2=1,124,4	SIM	1982
IF(X(I2,I3).EQ.VI*ES.AND.(I2+1,I3).EQ.VI*ES.AND.(I2+2,I3).EQ.	SIM	1983
1 NIKES.AND.(I2+3,I3).EQ.NIKES) GO TO 35	SIM	1984
GO 33 I1=1,4	SIM	1985
PHASE=Y(I1+I2-1,I3)	SIM	1986
IF(PHASE.EQ.NIKES) GO TO 82	SIM	1987
GO 31 I4=1,24	SIM	1988
COMRES(I4,I1)=ATN(PHASE,CKE)	SIM	1989
31 PHASE=5*ATN(PHASE,-1)	SIM	1990
GO TO 84	SIM	1991
32 DO 33 I4=1,24	SIM	1992
47 COMRES(I4,I1)=77777777	SIM	1993
34 CONTINUE	SIM	1994
X(I1+I2-1,5) NUL, CODE, (COMRES(25-I4,I1),I4=1,24),I1=1,4)	SIM	1995

GE/EE/72-7

55 CODE=CODE+349	SIN	1996
57 MVT=MVT+329	SIN	1997
WRITE(6,119)	SIN	1998
RETURN	SIN	1999
59 DO 97 I3=1,21	SIN	2000
CODE=7590	SIN	2001
90 95 I2=1,125,9	SIN	2002
IF(1-I2,I3).EQ.NINES.ENT.P(I2+1,I3).EQ.NINES.ENT.P(I2+2,I3).EQ.	SIN	2003
1 NINES.ENT.P(I2+3,I3).EQ.NINES.ENT.P(I2+4,I3).EQ.NINES.ENT.	SIN	2004
2 P(I2+5,I3).EQ.NINES.ENT.P(I2+6,I3).EQ.NINES.ENT.P(I2+7,I3)	SIN	2005
3 .EQ.NINES) GO TO 95	SIN	2006
90 93 I1=1,8	SIN	2007
IF(1-I1+I2-1,I3).EQ.1(MEC) GO TO 91	SIN	2008
GO TO 93	SIN	2009
91 P(I1+I2-1,I3)=7777777777	SIN	2010
CONTINUE	SIN	2011
WRITE(6,119) MVT,CODE,(M(I1+I2-1,I3),I1=1,3)	SIN	2012
95 CODE=CODE+169	SIN	2013
97 MVT=MVT+329	SIN	2014
WRITE(6,119)	SIN	2015
RETURN	SIN	2016
100 FORMAT(/ , " ** MEMORY DUMP **", //, " CHAN SECT", /)	SIN	2017
105 FC=AT(14,,14,02,37,07,34,21601,17,601,14,601,12,601,34)	SIN	2018
110 FC=AT(14,,14,02,37,07,34,41601,17,601,14,601,14,601,34)	SIN	2019
115 FORMAT(" ** END OF MEMORY DUMP **", 5X, " (POWERS OF MEMORY ACT 115	SIN	2020
120 CONTAIN TO INFORMATION PROVIDED BY THE PRESENT PROGRAM ONLY" /	SIN	2021
2 /)	SIN	2022
125 FC=AT(14,,14,02,37,07,4175,00)	SIN	2023
130 FC=AT(14,,14,02,37,07,4175,00)	SIN	2024
END	SIN	2025

SUPPLEMENTARY DISCREET	SIP	2526
IF (TESTED 1(24), CHAN, COME, COMREG(24,2), D3, E(3), EN, F(4), FC	SIP	2527
IF (TESTED 4(15), CH, CMT, PHASE, 2(24,4), REGIST, REGIST2, RI, FC	SIP	2528
IF (TESTED 5(13), SECT, SIGNAL, U, V(24,4), VI, WK, X(19,12), XI	SIP	2529
IF (TESTED 6(24,25), Y1, ZERO, DISPOSE	SIP	2530
COMMON 2, CHAN, CODE, COMREG, D3, E, EN, F, FC, M, I(24), IPES	SIP	2531
COMMON 1(24), 4(12,21), A(24), NCOL, LFLAG, MUP, X(19,12), PHASE	SIP	2532
COMMON 53, SIGNAL, 4, REGIST, REGIST2, RI, RC, SECT, C, V, VI, WK	SIP	2533
COMMON X, XI, Y, Y1, DISPOSE	SIP	2534
COMMON/COMMON 3, ON, ZERO, COM, KREG(15), TM(27), NBLANK, NCOMPI,	SIP	2535
COMMON/COMMON 4, LFLAG, NREPAIR, XRES	SIP	2536
ENTRY DISK	SIP	2537
XI=XI+1	SIP	2538
IF (XI,ST,13) GO TO 25	SIP	2539
1 NCOL=NCOL+1	SIP	2540
IF (NCOL,ST,72) RETURN	SIP	2541
IF (NCOL,ST,72) RETURN	SIP	2542
GO TO 1	SIP	2543
5 DO 15 I1=1,19	SIP	2544
X(25-I1,XI)=ZERO	SIP	2545
10 NCOL=NCOL+1	SIP	2546
IF (NCOL,ST,72) RETURN	SIP	2547
IF (NCOL,ST,72) RETURN	SIP	2548
1 NCOL=NCOL+1	SIP	2549
IF (NCOL,ST,72) RETURN	SIP	2550
25 CONTINUE	SIP	2551
20 NCOL=NCOL+1	SIP	2552
IF (NCOL,ST,72) RETURN	SIP	2553
IF (NCOL,ST,72) RETURN	SIP	2554
GO TO 20	SIP	2555
25 WRITE(6,115)	SIP	2556
WRITE(4,115)	SIP	2557
GO TO 20	SIP	2558
ENTRY DISK	SIP	2559
YI=YI+1	SIP	2560
IF (YI,ST,13) GO TO 51	SIP	2561
20 NCOL=NCOL+1	SIP	2562
IF (NCOL,ST,72) RETURN	SIP	2563
IF (NCOL,ST,72) RETURN	SIP	2564
GO TO 20	SIP	2565
35 DO 45 I2=1,24	SIP	2566
Y(25-I2,YI)=ZERO	SIP	2567
40 NCOL=NCOL+1	SIP	2568
IF (NCOL,ST,72) RETURN	SIP	2569
IF (NCOL,ST,72) RETURN	SIP	2570
1 NCOL=NCOL+1	SIP	2571
IF (NCOL,ST,72) RETURN	SIP	2572
45 CONTINUE	SIP	2573
GO TO 20	SIP	2574
50 WRITE(6,115)	SIP	2575
WRITE(4,115)	SIP	2576
GO TO 20	SIP	2577
115 FORMAT(' THE SYMBOL "0,21," IS NOT A VALID X-DISCRETE SIGNAL. IT	SIP	2578
1 HAS BEEN REPLACED BY ZERO')	SIP	2579
125 FORMAT(' MAXIMUM ALLOWED Y-DISCRETE STORAGE REQUESTS IS 1 - THESE	SIP	2580
13 REQUEST NOT ACCEPTED')	SIP	2581
117 FORMAT(' THE SYMBOL "0,41," IS NOT A VALID Y-DISCRETE SIGNAL. IT	SIP	2582
1 HAS BEEN REPLACED BY ZERO')	SIP	2583
119 FORMAT(' MAXIMUM ALLOWED Y-DISCRETE STORAGE REQUESTS IS 12 - THESE	SIP	2584
120 REQUEST NOT ACCEPTED')	SIP	2585
END	SIP	2586

SUBROUTINE INCREP	SIP	2387
INTEGER 4(24), CM, CODE, COMSEC(24,2), CO, E(2), EN, F(4), FC	SIP	2388
INTEGER 4(15), CH, CSE, PHASE, R(24,4), REGIST, REGISTB, CI, CK	SIP	2389
INTEGER 5(2), SECT, SIGNAL, U, V(24,4), VI, VC, X(19,19), XI	SIP	2390
INTEGER V(24,19), YI, ZERC, DISPOSE	SIP	2391
COMMON 1, CM, CODE, COMSEC, CO, E, EN, F, FC, U, I(24), IREG	SIP	2392
COMMON 1(24), 4(128,21), 8(24), NCCL, NFLAG, NUP, NUP2(72), PHASE	SIP	2393
COMMON 53, SIGNAL, R, REGIST, REGISTB, RI, RC, SECT, U, V, VI, VC	SIP	2394
COMMON 1, XI, Y, YI, DISPOSE	SIP	2395
COMMON/REGIST/ CM, ZERC, COM, KRES(19), M(27), NFLAG, NCORNA,	SIP	2396
ALPACR, ALPACR, NLAGS	SIP	2397
1 ENTRY INCV	SIP	2398
RI=RI+1	SIP	2399
IF(RI.GT.4) RI=1	SIP	2400
1 NCCL=NCCL+1	SIP	2401
IF(NCCL.GT.72) RETURN	SIP	2402
IF(MOD2(NCCL).EQ.NUPACR) GO TO 5	SIP	2403
GO TO 1	SIP	2404
5 DO 15 I1=1,24	SIP	2405
R(25-I1,RI)=ZERC	SIP	2406
10 NCCL=NCCL+1	SIP	2407
IF(NCCL.GT.72) RETURN	SIP	2408
IF(MOD2(NCCL).EQ.NUPACR) GO TO 15	SIP	2409
IF(MOD2(NCCL).EQ.NFLAG) GO TO 15	SIP	2410
IF(MOD2(NCCL).EQ.NUPACR) GO TO 15	SIP	2411
1 NUP2(NCCL)	SIP	2412
IF(MOD2(NCCL).EQ.NUPACR) R(25-I1,RI)=COM	SIP	2413
15 CONTINUE	SIP	2414
25 SECT=RI	SIP	2415
REGIST=REG(12)	SIP	2416
20 IF(REGIST.EQ.0) GO TO 25	SIP	2417
CALL REG2	SIP	2418
25 NCCL=NCCL+1	SIP	2419
IF(NCCL.GT.72) RETURN	SIP	2420
IF(MOD2(NCCL).EQ.NUPACR) RETURN	SIP	2421
GO TO 25	SIP	2422
ENTRY INCV	SIP	2423
VI=VI+1	SIP	2424
IF(VI.GT.4) VI=1	SIP	2425
70 NCCL=NCCL+1	SIP	2426
IF(NCCL.GT.72) RETURN	SIP	2427
IF(MOD2(NCCL).EQ.NUPACR) GO TO 25	SIP	2428
GO TO 25	SIP	2429
75 DO 45 I1=1,24	SIP	2430
V(25-I1,VI)=ZERC	SIP	2431
40 NCCL=NCCL+1	SIP	2432
IF(NCCL.GT.72) RETURN	SIP	2433
IF(MOD2(NCCL).EQ.NUPACR) GO TO 45	SIP	2434
IF(MOD2(NCCL).EQ.NFLAG) GO TO 45	SIP	2435
IF(MOD2(NCCL).EQ.NUPACR) GO TO 45	SIP	2436
1 NUP2(NCCL)	SIP	2437
IF(MOD2(NCCL).EQ.NUPACR) V(25-I1,VI)=COM	SIP	2438
45 CONTINUE	SIP	2439
46 SECT=VI	SIP	2440
REGIST=REG(19)	SIP	2441
GO TO 25	SIP	2442
238 FORMAT(' THE SYMBOL "0,01,0" IS NOT A VALID R-INDEPENDENT INPUT.	SIP	2443
237 WAS BEEN REPLACED BY ZERO')	SIP	2444
235 FORMAT(' THE SYMBOL "0,01,0" IS NOT A VALID V-INDEPENDENT INPUT.	SIP	2445
237 WAS BEEN REPLACED BY ZERO')	SIP	2446
END	SIP	2447

GE/EE/72-7

Appendix B

D17B Instruction Set

and

D17B Load Codes

D178 Instruction Set

CODE	DESCRIPTION	NUMERIC CODE	WORD TIMES
ADD	Add	64 0,s	1
ALS	Accumulator Left Shift	00 22,s	s+1
ANA	Logical And to Accumulator	40 42,s	1
ARS	Accumulator Right Shift	00 32,s	s+1
BOA	Binary Output A	40 10,s	1
BOB	Binary Output B	40 12,s	1
BOC	Binary Output C	40 02,s	1
CLA	Clear and Add to Accumulator	44 c,s	1
COA	Character Output A	00 40,s	s+1
COM	Complement	40 46,s	1
DIA	Discrete Input A	40 52,s	1
DIB	Discrete Input B	40 50,s	1
DOA	Discrete Output A	40 26,s	1
EFC	Enter Fine Countdown	40 62,s	1
HFC	Halt Fine Countdown	40 60,s	1
HPC	Halt and Proceed	40 22,s	1
LFR	Load Phase Register	40 7-,s	1
MIN	Minus Magnitude	40 44,s	1
MPX	Multiply Modified	34 c,s	13
KPY	Multiply	24 c,s	13
RSD	Reset Detector	40 20,s	1
SAD	Split Add	60 c,s	1
SAL	Split Accumulator Left Shift	00 20,s	s+1
SAR	Split Accumulator Right Shift	00 30,s	s+1
SCL	Split Compare and Limit	04 c,s	2
SLL	Split Left Word Left Shift	00 24,s	s+1
SLR	Split Left Word Right Shift	00 34,s	s+1
SMX	Split Multiply Modified	30 c,s	7
SNP	Split Multiply	20 c,s	7
SRL	Split Right Word Left Shift	00 26,s	s+1
SRR	Split Right Word Right Shift	00 36,s	s+1
SSU	Split Subtract	70 c,s	1
STC	Store Accumulator	54 c,s	1

D17B Instruction Set (cont'd)

<u>CODE</u>	<u>DESCRIPTION</u>	<u>NUMERIC CODE</u>	<u>WORD</u>	<u>TIME</u>
SUB	Subtract	74 c,s		1
THI	Transfer on Minus	10 c,s		1
TRA	Transfer	50 c,s		1
VOA	Voltage Output A	40 30,s		1
VOB	Voltage Output B	40 32,s		1
VOC	Voltage Output C	40 34,s		1

D17B Load Codes

- HALT** - This load code causes the D17B to stop accepting data and enter the program halt mode. The manual halt mode will be entered if the compute switch is set at the Halt position.
- COMPUTE** - This code causes the computer to go to the manual halt mode of noncompute. The compute mode will be entered if the compute switch is set at the Run or Single position.
- FILL** - This load code "0" sets the fill/verify flipflop (0_3).
- VERIFY** - This load code "1" sets the fill/verify flipflop (0_3).
- LOCATE** - This code causes the contents of the Lower Accumulator to be shifted into the Instruction Register.
- CLEAR** - This code clears the Lower Accumulator by filling it with all zeros.
- DELETE** - This code causes no action.
- ENTER** - The action produced by this code depends upon the setting of the fill/verify flipflop. When this code is first deciphered, the contents of the Lower Accumulator is transferred into the Accumulator.

If O_3 is "0" set, then the contents of the Accumulator is stored in the memory location addressed by the Instruction Register.

If O_3 is "1" set, then the contents of the Accumulator is compared with the contents of the memory location addressed by the Instruction Register.

The last action of this load code is to increment the Instruction Register by one.

Appendix C

Figures for Interpreting -
Binary, Discrete, and Voltage Outputs

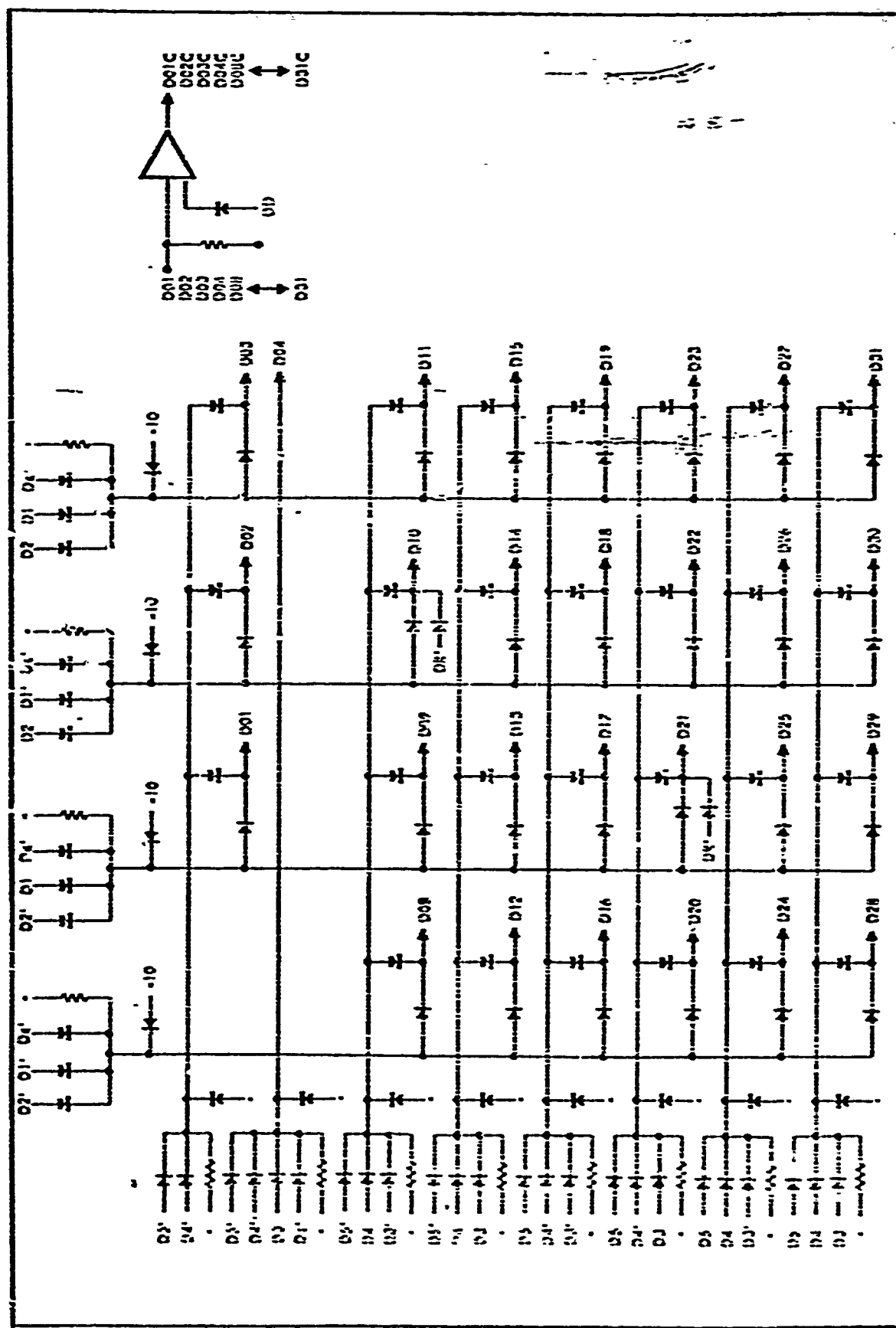


Fig. 12. Discrete Outputs (Ref 4:TR-8)

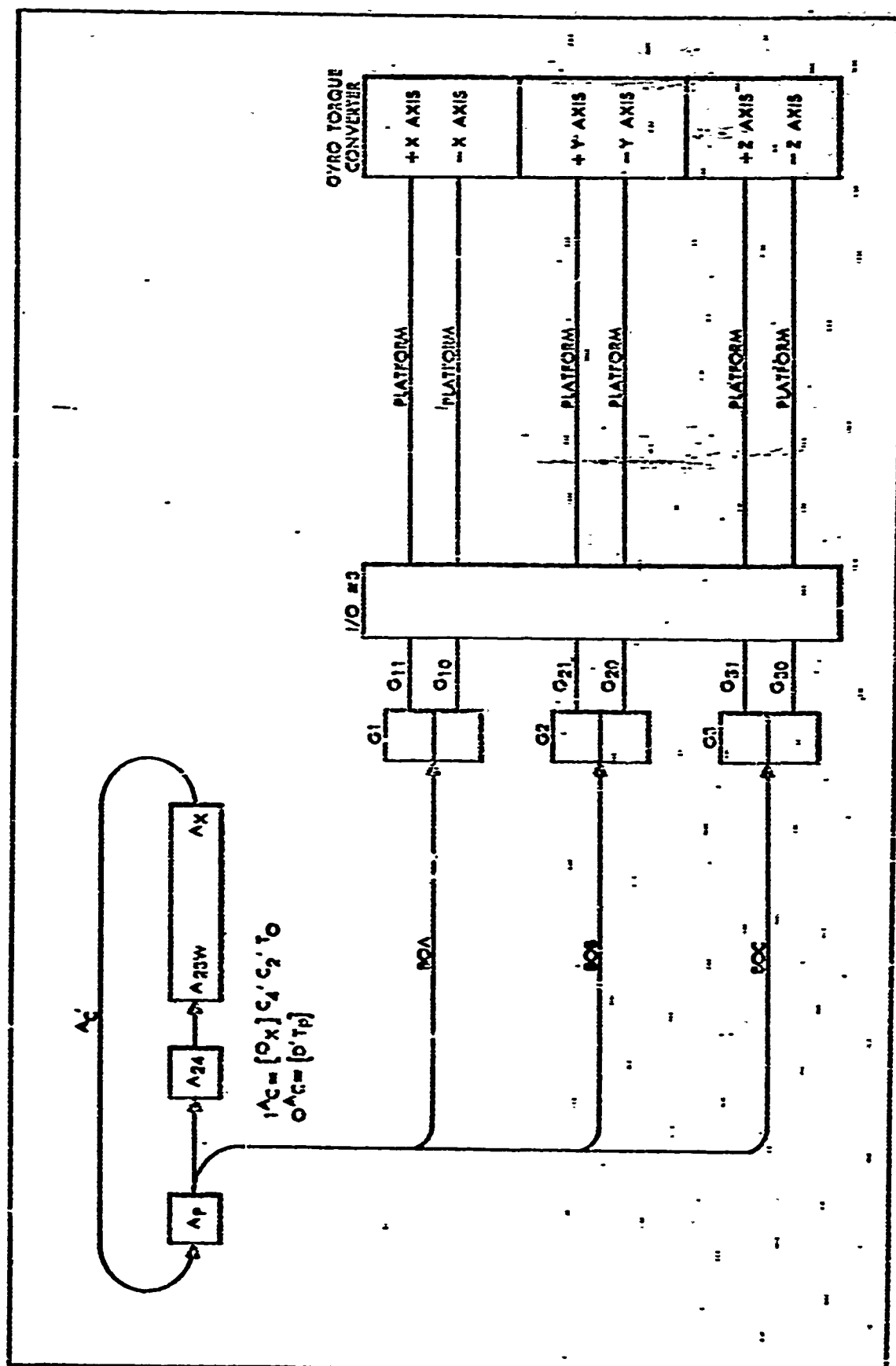


Fig. 13. Binary Outputs (Ref 5:57)

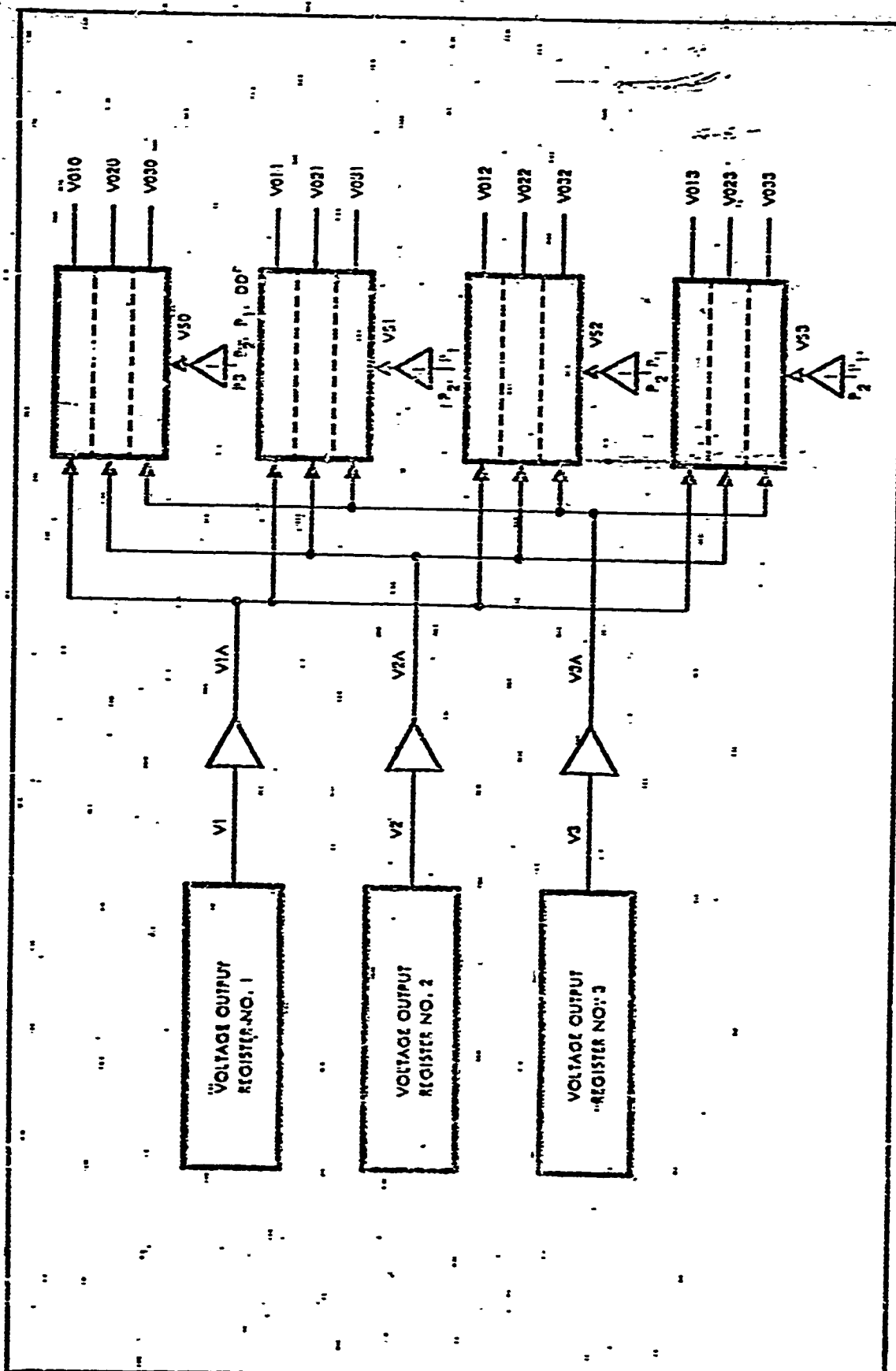


Fig. 14. Voltage Output Scheme (Ref 5:56)

GE/EE/72-7

Appendix D

Instructions for Using
the D17B Computer Simulation Program
at AFIT

Foreword to Appendix D

A software simulation program has been written which simulates the functions of the Minuteman D17B computer. The structure and organization of this simulation program is described in chapter II of this thesis. The D17B simulation language is presented in chapter III, chapter IV contains a listing of the error statements of the simulation program, and chapter V is made up of example programs which were run on the simulated computer.

This appendix contains information for using the D17B computer simulation program at AFIT. Procedures are given for accessing the simulation program from a teletype terminal. Information concerning the use of program tapes and external program files is also included. A condensed version of the D17B simulation language is given followed by the listing of a method for creating a shortened version of the simulation language.

Procedures for Using the
D17B Computer Simulation Program at AFIT

The D17B computer simulation program was written to be used from a teletype terminal. Procedures for operating a teletype terminal are contained in the Intercom 2 Reference Manual (Ref 3).

The simulation program is available on the CDC 6600 Computer System as a permanent file. MHSIMULATION is the permanent file name. Only one version of the program exists so a cycle number need not be specified when attaching the permanent file. However, the program was catalogued as CY=1. The simulation program requires less than 40% of memory to execute on the 6600 system. The majority of programs run on the simulation program require less than 5 seconds of central processor time.

Operation from a Teletype Terminal. To access the simulation program from the teletype terminal requires the user to LOGIN with the 6600 computer. Procedures for doing this are contained in the Intercom 2 Reference Manual (Ref 3: Chap. 3, p. 3). After login has been successfully completed the teletype prints out

COMMAND-

The user will respond with

ATTACH, MHSIMULATION, CY=1.

to make the simulation program available for his use. The system will respond to this request by typing the time and

the attach request followed by

COMMAND-

The user must decide if he wants the output to be printed at the teletype or if he wants to dispose the output to the batch terminal line printer. If output is to be printed by the teletype the user should respond to the command with

CONNECT, INPUT, OUTPUT.

If output is to be disposed to the high speed printer the user should respond with

CONNECT, INPUT, TAPE4.

The system will process this request and the teletype will print the following:

COMMAND-

The user should respond with

HAHA.

which puts the simulation program into execution. During execution of his data the user should respond to any messages that appear at the teletype. When the user has finished running programs he should exit the simulation program by specifying PE(OFF) and respond to the message:

"TO RUN ANOTHER PROGRAM TYPE 'RUN'; TO STOP TYPE

'HALT' -

by typing

HAHA

The system responds with the message

"END OF PROGRAM"

and prints the amount of execution time used. This will be followed by

COMMAND-

If output is to be disposed to the batch terminal line printer, the user should respond with

DISPOSE, OUTPUT, P2-ET7.

followed by the LOGOUT procedures contained in the Intercom 2 Reference Manual (Ref 3:Chap. 3, p. 3-6).

If output is not to be disposed to the line printer, then the user should follow the procedures for LOGOUT.

Using Program Tapes. Program tapes can supply data to the simulation program. The data on these tapes is entered by the tape reader located at each teletype terminal. The program tapes can also be punched at the teletype terminal. Procedures for punching and reading of program tapes is contained in the Intercom 2 Reference Manual (Ref 3:Chap. 2, P. 7-9).

Program tapes which will also be run on the D173 computer must be written using the ASCII representation described in chapter III of this thesis. Blanks, line feeds, and carriage returns are ignored by the D173 computer tape reader, so these symbols can be punched on the tapes. This allows the program tapes which will be run on the D173 computer to also be executed by the simulation program.

Using External Files. Two files can be established in the CDC 6600 computer which will supply data in the ASCII representation to the simulation program. The files must have the names of TAPE2 or TAPE3. They can be created at the teletype by entering SETUP when the teletype prints

COMMAND-

The user responds with

SETUP.

The teletype will process this command and print back

NEW OR OLD FILE--

The user should type

NEW/TAPE2 or NEW/TAPE3

The teletype will respond with

READY.

The user can proceed to write a program in ASCII representation. Each line of the program must be preceded by a line number. Procedures for creating programs in SETUP are contained in the INTERCON 2 Reference Manual (Ref 3:Chap. 4). The last symbol supplied must be the letter "H". This symbol signifies the end of the program. Also the SAVE directive should always be given at the end of a program.

To use the program created on TAPE2 or TAPE3, the user must provide an argument to the simulation program command KMAN. KMAN(2) will result in the reading of the data on TAPE2. KMAN(3) causes the simulation program to read the data from TAPE3. An example in which data is read from TAPE2 is as follows:

(ENTER PROGRAM)

PR(ON) MR(ON) EM(ON) PS(ON) MMAN(2)

MR(ON) K(RUN)

PR(OFF)

D17B Counter Simulation Program Language. A listing of the simulation language is as follows:

Octal numbers - 0, 1, 2, 3, 4, 5, 6, 7

Binary numbers - 0, 1

Load Codes - HALT, LOCATION, FILL, VERIFY
COMPUTE, ENTER, CLEAR, DELETE

When OCTAL is specified, input must be in Octal representation. When BINARY is specified, input must be in Binary representation. When MMAN is specified, input must be in ASCII representation. (Default is OCTAL)

	<u>Octal Representation</u>	<u>Binary Representation</u>	<u>ASCII Representation</u>
Numbers -	0	10000	0
	1	00001	1
	2	00010	2
	3	10011	3
	4	00100	4
	5	10101	5
	6	10110	6
	7	00111	7
Load Codes -			
HALT		01000	8
LOCATION		11001	9
FILL		11010	Z
VERIFY		01011	;
COMPUTE		11100	<
ENTER		01101	=

CLEAR	01110	^
DELETE	11111	?

<u>Switch Name</u>	<u>Mnemonic(Setting)</u>
Switches - Timing Signal	T(ON)
Power On/Off Switch	PR(ON), PR(OFF)
Initiate Load Switch	PS(ON)
Master Reset Switch	MR(ON)
Cold-Storage Write Switch	EW(ON), EW(OFF)
Discrete Switch	DD(ON), DD(OFF)
Mechanical Input Switch	IN(ON)
(Default of these switches is OFF)	
Compute Mode Switch	K(HALT), K(SINGLE), K(RUN)
(Default of this switches is HALT)	

Display - A binary output listing of any of the following registers or loops will be given whenever its contents changes, if it appears as the argument of the REGISTER command:

<u>Mnemonic</u>	<u>Register or Loop</u>
A	Accumulator
I	Instruction Register
L	Lower Accumulator
N	Number Register
P	P-loop
E	E-loop
H	H-loop
U	U-loop
V	V-loop
R	R-loop

Example: REGISTER(A,I,L,N)
REGISTER(AILNEPUEVR)

(No. of arguments can vary from 0 to 10 and must be one of those given above)

The contents of memory will be given as output whenever a MEMORY command is given:

MEMORY(OCTAL) Memory dump will be given in Octal.

MEMORY(BINARY) Memory dump will be given in Binary.

(Default is OCTAL)

Discrete Inputs - X(19 bits as argument)

Y(24 bits as argument)

Example: X(1 000 000 101 111 110)

Y(111000111000111000111000)

Incremental Inputs - V(24 bits as argument)

R(24 bits as argument)

Example: V(0000 0001 0010 0011 0100 0101)

R(00001111 11110000 10101010)

Miscellaneous -

SIGNAL - Each time SIGNAL is used, it flips the representation from 0 to 1 or 1 to 0. SIGNAL is 0 at the beginning of the program run. When SIGNAL is 1, the modes of the computer will be traced.

EXECUTE(XXX) - No. of execution cycles in the compute mode can be specified.

Example: EXECUTE(0010) 10 executions

EXECUTE(9999) 9999 executions

EXECUTE(0250) 250 executions

(Default is 50 executions)

DR - DR flipflop is "1" set.

VK(0) - VK flipflop is "0" set.

VK(1) - VK flipflop is "1" set.

RK(0) - RK flipflop is "0" set.

RK(1) - RK flipflop is "1" set.

REINITIALIZE - Memory is initialized, binary output flipflops are set to +1, and discrete input counters are set to zero.

Shortened Version of Simulation Language. The following listing contains the simulation language words which can be shortened, the interpreting letters, and an example of a shortened version:

<u>Simulation Language words which can be Shortened</u>	<u>Interpreting Letters</u>	<u>Example of a Shortened Version</u>
HALT	H	HALT
LOCATION	L	LCC
FILL	FI	FILL
VERIFY	V	VER
COMPUTE	CO	COM
ENTER	EN	EN
CLEAR	CL	CL
DELETE	DE	DEL
OCTAL	O	OCT
BINARY	B	BIN
HEXAN	HE	HEXAN
SINGLE	S	SING
RUN	R	RUN
REGISTER(Arg)	RE(Arg)	REG(Arg)
MEMORY(BINARY)	ME(B)	MEM(B)
MEMORY(OCTAL)	ME(O)	MEM(O)
SIGNAL	S	SIG
EXECUTE(Arg)	EX(Arg)	EXEC(Arg)
REINITIALIZATION	REI	REINIT